ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

Simulador de Fútbol Online desarrollado bajo Ruby On Rails

Carlos Rafael Belizón Ibáñez

4 de septiembre de 2012

ESCUELA SUPERIOR DE INGENIERÍA

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

SIMULADOR DE FÚTBOL ONLINE DESARROLLADO BAJO RUBY ON RAILS

- Departamento: Ingeniería Informática
- Directores del proyecto: Juan Manuel Dodero Beardo y Manuel Palomo Duarte
- Autor del proyecto: Carlos Rafael Belizón Ibáñez

Cádiz, 4 de septiembre de 2012

Fdo: Carlos Rafael Belizón Ibáñez

Agradecimientos

Me gustaria agradecer y dedicar este texto a mis padres y hermano por la paciencia que han tenido, así como a mis amigos que siempre han estado animándome para finalizarlo.

Licencia

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

1.1. Introducción

Este documento describe el proceso que hemos seguido para la realización del proyecto. Comenzamos con la recopilación de requisitos obteniendo al tinal un documento de Especifacación de Requisitos del Software, realizada a través de las sucesivas entrevistas con los tutores del proyecto de fin de carrera.

Posteriormente seleccionamos la tecnología más adecuada para llevar a cabo el proyecto. A continuación pasamos a desarrollar los componentes del software que añadiremos al sistema para que cumpla con los requisitos que hemos recopilado previamente.

Para finalizar, seguiremos un desarollo software ágil, por el cual vamos desarollando partes del sistema de forma que sean totalmente funcionales una vez terminadas hasta conformar el proyecto según las especificaciones que se han recogido.

1.2. Objetivos

Los objetivos de este proyecto son por una parte poner en práctica los conocimientos adquiridos a lo largo de la formación académica recibida en la titulación de *Ingeniería Técnica en Informática de Gestión* y a la vez aprender a desarrollar bajo una novedosa tecnología web basada en un *framework* basado en el patrón Modelo-Vista-Controlador (***MVC***) denominado *Ruby On Rails* (***MVC***).

Este *framework*, a su vez, basa su simplicidad en el uso de un lenguaje de *script* llamado *Ruby* que tiene características propias de lenguajes funcionales; además, es uno de los exponentes de una perspectiva de desarrollo en auge que se engloba dentro de las denominadas metodologías ágiles.

El resultado de este proyecto es el desarrollo de una aplicación informática *web* de código abierto para realizar la simulación virtual de la *gestión computerizada* de un club de fútbol, la simulación de los partidos que se jueguen, además de la interacción con otros usuarios. En definitiva: un videojuego multijugador con interfaz web.

Pasamos a enumerar los objetivos de forma esquemática:

- **Gestión de usuarios:** todas las herramientas necesarias para identificar a cada usuario de nuestra aplicación web de forma unívoca.
- **Gestión de clubes:** todas las herramientas necesarias para gestionar precios de entrada, venta, compra, entrenamiento de futbolistas y tipos de entrenamientos.
- **Gestión de campeonatos:** todas las herramientas necesarias para que el administrador pueda controlar el calendario de competición, y la creación es de nuevas ligas en función de los usuarios que se vayan registrando en la web.
- **Simulación de partidos:** desarrollaremos un simulador que mediante diferentes ponderaciones sea capaz de simular la realidad de un partido de fútbol 11.

1.3. Alcance

Se desea informatizar toda la gestión y simulación de un entorno de competición de fútbol 11 de forma que acabemos obteniendo un videojuego multijugador interactivo con interfaz web.

1.3.1. Identificación del producto mediante un nombre

El producto realizado se llama “Football Simulator”.

A lo largo de toda la memoria nos referiremos a él como ***Simulador de fútbol online***.

1.3.2. Funcionalidades del producto

El producto que deseamos obtener tal y como hemos mencionado anteriormente es capaz de simular la gestión, interacción y desarrollo de una competición de liga de fútbol 11, proporcionando funcionalidades de interacción entre usuarios y del administrador para controlar el comportamiento de la competición.

Por otra parte, nuestro producto también es capaz de simular mediante diversas ponderaciones los resultados de un partido de fútbol.

Para disfrutar de las funcionalidades de nuestro producto es necesario el registro y participación de varios usuarios en él, además de la interacción de un administrador que llevará a cabo acciones de control, supervisión y seguimiento del desarrollo de la competición.

1.3.3. Aplicaciones del software: beneficios, objetivos y metas

Con este producto podemos obtener diversas ventajas ya descritas anteriormente, entre ellas cabe destacar la gestión de los equipos de fútbol y la interacción de usuarios.

El principal objetivo de la utilización del producto es divertir y entretener al usuario, mediante la inmersión en un entorno virtual de gestión de clubes con diversos usuarios que compiten entre sí para alcanzar la posición más alta en clasificación y división que puedan.

Al fin y al cabo lo que se desea es formar la comunidad más grande posible y para ello es necesario hacer que el usuario desee acceder asiduamente a nuestra aplicación.

La mayor ventaja es que al ser una aplicación web, cualquier usuario puede registrarse sin ningún tipo de restricción.

1.4. Definiciones, acrónimos y abreviaturas

- **Framework:** Estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de sofware concretos, con base en la cual otro proyecto software puede ser organizado y desarrollado.
- **Patrón Modelo Vista Controlador (MVC):** Es un estilo de arquitectura software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.
- **COC (Convenction over Configuration):** En español significa *"Convención sobre configuración"*, quiere decir que toda pieza de código tiene su lugar y todas ellas interactúan según un camino estándar.
- **DRY (Don't Repeat Yourself):** En español significa *"No te repitas"*. Se refiera a que es innecesario repetir código si éste está bien diseñado y refactorizado.
- **Programación Orientada a Objetos (POO):** . Es un paradigma de la programación que se basa en la utilización de clases de objetos.
- **Simulador de Fútbol Online:** es el nombre utilizado comúnmente durante todo el documento para hacer referencia al nombre del proyecto.
- **Ruby:** Lenguaje de programación orientado a objetos de script con características propias de lenguajes funcionales.
- **Gema:**
- **Ruby On Rails (RoR):** Framework de desarrollo rápido de aplicaciones web basado en el lenguaje de programación *Ruby*.
- **GIT:** Sistema de control de la configuración distribuido.
- **Integrated Development Environment (IDE):** En español significa *"Entorno integrado de desarrollo"*. Es un programa informático compuesto por un conjunto de herramientas y módulos de programación que ayudan al desarrollo de productos software.

- **NetBeans:** IDE con funcionalidades que ayudan al desarrollo de aplicaciones *Ruby on Rails*
- **Sistema de Gestión de Base de Datos (SGBD)**:
- **Oracle:** Es un sistema de Gestión de Base de Datos privativo.
- **MySQL:** Sistema de gestión de base de datos de software libre.
- **SQLite2:** Sistema de gestión de base de datos de software libre.
- **SQLite3:** Sistema de gestión de base de datos de software libre.
- **Postgre:** Sistema de gestión de base de datos de software libre.
- **Cascade Style Sheet (CSS):** En español significa *"Hojas de estilo en cascada"*.
- **Hypertext Markup Language (HTML))** En español significa *"Lenguaje de marcado de hipertexto"*.
- **JavaScript:** Lenguaje de script que se ejecuta en el lado cliente.
- **JQuery:** Librería de desarrollo rápido JavaScript.
- **Algoritmo criptográfico:** En computación y criptografía un algoritmo criptográfico es un algoritmo que modifica los datos de un documento con el objeto de alcanzar algunas características de seguridad como autenticación, integridad y confidencialidad.
- **Algoritmo MD5:** En criptografía, MD5 (abreviatura de Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5) es un *algoritmo criptográfico* de 128 bits ampliamente usado.
- **Log:** Archivo de texto plano que nos indica las todas las acciones que se realizan en nuestra aplicación con la finalidad de conseguir auditar el uso de la misma.
- **Estándar W3C**: Cuando se describe que un sitio o página web cumplen con ciertos estándares web, usualmente quiere decir que la página tiene partes de código HTML, CSS y JavaScript válido o casi válido.

1.5. Visión General

Esta memoria sigue las pautas descritas en el documento de referencia *"Recomendaciones para la documentación del Proyecto de fin de Carrera"* realizada por varios profesores del departamento de lenguajes y sistemas informáticos.

Las partes que componen el documento son:

1. **Introducción:** Visión general. Objetivos, alcance y estructura.
2. **Planificación:** Planificación temporal del desarrollo del proyecto y porcentaje de esfuerzo dedicado a las diversas tareas y fases que lo forman.
3. **Descripción general del proyecto:** Ampliación de la visión global del proyecto. Perspectiva, funciones y restricciones.

4. **Análisis:** Identificación de las metas globales, perspectivas del cliente y recogida de información necesaria para llevar a cabo el proyecto.
5. **Diseño:** Aplicamos diferentes técnicas y procedimientos para obtener un producto con el suficiente detalle que permita su realización física.
6. **Implementación:** Aspectos más relevantes de la implementación de la aplicación software.
7. **Pruebas:** Todo lo concerniente a las pruebas para que la aplicación logre su cometido sin fallos.
8. **Resumen:** Breve resumen de lo más destacable del proyecto realizado.
9. **Conclusiones y trabajo futuro:** Valoración global, posibles mejoras y ampliaciones del proyecto.
10. **Apéndices:**
    - **Manual de usuario:** Manual del funcionamiento de la aplicación destinado a los usuarios de la misma.
    - **Manual de instalación y puesta en funcionamiento:** Instrucciones para realizar la instalación de la aplicación, así como su puesta en funcionamiento.
    - **Difusión:** Lugares dónde se ha publicado el proyecto.
    - **Licencia:** Texto de la licencia adoptada para la divulgación del proyecto.
11. **Bibliografía:** Libros y referencias físicas y electrónicas consultadas para la elaboración del proyecto.

CAPÍTULO 2

Planificación

2.1. Metodología de desarrollo

La distribución temporal seguida en líneas generales es la descrita a continuación:

- Análisis: En esta fase llevaremos a cabo la recogida de todos los requisitos funcionales de nuestra aplicación. Al utilizar una metodología ágil la recogida de requisitos podrá ser más laxa que si hubiéramos seguido un modelo lineal.
- Diseño: En esta fase recrearemos diseños conceptuales de cómo serán nuestras vistas, y a partir de ellas podremos obtener nuestros modelos de datos.
- Programación: En esta fase nos dedicaremos a codificar de forma sinérgica las diferentes secciones en las que se compone un desarrollo basado en el patrón Modelo Vista Controlador
- Pruebas: Las pruebas que realizaremos serán:
    1. Test de usabilidad
    2. Pruebas de caja negra.
    3. Estudio y resolución de los resultados de las pruebas.
- Documentación: Para la documentación presentaremos un manual de usuario y de instalación detallado.

2.2. Calendario

En la figura 2.1 se puede ver un diagrama de Gantt con la división y el tiempo tomado para realizar cada una de las tareas en las que hemos dividido el proyecto. La unidad de tiempo básica está expresada en días laborables de 8 horas.

Aunque la etapa de Memoria se ha realizado de forma solidaria con las demás, para simplificar se ha extraído en un sólo bloque.

- Entrevistas: Nos reunimos con nuestros tutores de proyecto para recopilar los requisitos que deberá tener nuestro proyecto de fin de carrera para que sea lo suficientemente completo para pasar la aprobación del tribunal y aprender de forma exhausta el framework *Ruby On Rails* (5 días).
- Análisis: Una vez recopilada toda la información en las entrevistas con nuestros tutores, extraemos la información importante y la esquematizamos (7 días).
- Especificación de Requisitos: Definimos, organizamos, cada uno de los requisitos que hayamos extraído y los desarrollamos de forma exhausta. En esta fase también describiremos los casos de uso, evitando en la medida de lo posible la redundancia de los mismos y maximizando la posibilidad de reutilizarlos (10 días).
- Diseño: En esta fase estudiaremos la mejor forma para incorporar la funcionalidad del sistema teniendo en cuenta que usaremos el framework de desarrollo rápido de aplicaciones *Ruby On Rails*, con especial atención a que haremos uso del patrón Modelo-Vista-Controlador (25 días).
- Implementación: En esta fase codificaremos la aplicación para que sea funcional según hemos especificado en la fase de diseño; además, será necesario aprender la forma de utilizar de forma correcta el framework *Ruby On Rails)*. En esta fase va incluida la creación de la estructuras necesarias para la interacción con el Sistema de Gestión de Bases de Datos (*SGBD*) a través de un sistema *ORM* (90 días).
- Pruebas: En este paso comprobaremos que la corrección del sistema que hemos desarrollado. En caso de encontrar algún fallo es el momento de solucionarlo y repetir las pruebas (5 días).
- Memoria: Toda la documentación que conforma esta memoria es el resultado de detallar cada uno de los documentos que hemos ido generando a través de la realización de la misma (40 días).
- Presentación: Creación y preparación de la presentación que mostraremos en la etapa de defensa ante el tribunal (10 días).

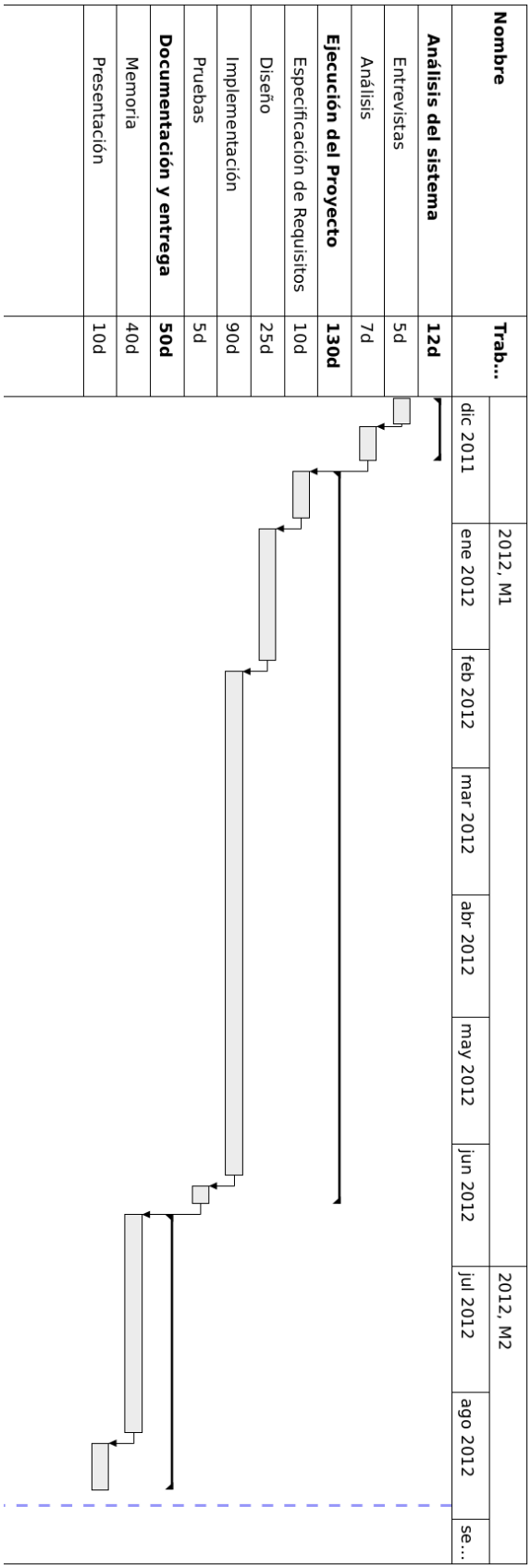

Figura 2.1: Diagrama de Gantt

2.3. Presupuesto

Para realizar una estimación del montante económico del proyecto realizado será necesario calcular y dirimir todos los recursos que han sido necesarios para llevarlo a cabo.

En dicho presupuesto se incluirán los materiales de oficina como serían escritorio, folios, ordenador, impresora, y demás recursos ofimáticos. Dichos costes los estimaremos en un 10 % del gasto final de nuestra aplicación.

Sabiendo que incluiremos todos los gastos que podríamos considerar como indirectos, para calcular el coste de mantener a un trabajador, tendremos que fijarnos en el gasto que supondría el coste de mantener a un *Analista-programador y Diseñador de página Web* en la empresa.

Para ello consultaremos las tablas salariales de dicha categoría profesional en el *XVI CONVENIO COLECTIVO ESTATAL DE EMPRESAS CONSULTORAS, DE PLANIFICACIÓN, ORGANIZACIÓN DE EMPRESA Y CONTABLE, EMPRESAS DE SERVICIOS DE INFORMÁTICA Y DE ESTUDIOS DE MERCADO Y DE LA OPINIÓN PÚBLICA* tal y como vemos en la tabla 2.1. Para el cálculo del pago de la Seguridad Social aplicaremos un 33 % del coste del salario bruto del trabajador.

Nombre	Coste mensual	Coste anual	Coste jornada (25 días por mes)
Analista-programador y Diseñador Web	1.468,54 €	20.559,56	58,75 €
Seguridad Social	484,82 €	6.784,65 €	No aplica

Tabla 2.1: Salario según último convenio de las TIC

Teniendo en cuenta que el proyecto lo hemos realizado en solitario y nos ha llevado un total de 192 jornadas laborales, lo que equivaldría a 7 meses y 3 semanas naturales, tendremos que el coste total del proyecto sin incluir los costes indirectos es el referido en la tabla 2.2

Cantidad	Descripción	Coste unitario	Coste Total
192	Jornada Analista-Programador y Diseñador Web	58,75 €	11.280 €
8	Seguridad Social Analista-Programador y Diseñador Web	484,82 €	3.878,56 €
		Total	15.158,56 €

Tabla 2.2: Coste total sin gastos indirectos del proyecto realizado

Si le añadimos el 10 % que representan los costes indirectos de la realización de nuestro proyecto, tendremos que el coste final real es el referido en la tabla 2.3

Descripción	Coste
Coste total	15.158,56 €
Costes indirectos	1.515,85 €
Total	16.674,41 €

Tabla 2.3: Coste total del proyecto realizado

CAPÍTULO 3

Descripción General del proyecto

3.1. Perspectiva del Producto

En este capítulo explicaremos qué pretendemos conseguir con la realización de este proyecto y qué características generales ha de cumplir.

3.1.1. Dependencias del producto

El proyecto es independiente, no es un subsistema de un proyecto de mayor envergadura, tampoco es continuación de ningún otro proyecto, pero al estar liberado con una licencia de *software libre* podría ser continuado, mejorado y actualizado por su propio creador o terceras personas ajenas a éste.

3.1.2. Interfaces de usuario

La interfaz de usuario se basa en una página web dinámica que lleva asociada varias opciones para poder usar las herramientas proporcionadas.

Para el uso de la aplicación, la página principal dispone de varios menús, entre ellos la de selección de idiomas y un *hipervínculo* al subsistema de registro e inicio de sesión.

Todo el diseño de la aplicación respeta los estándares *HTML 4.0* y *CSS 2.0* así como el uso de la librería de *JavaScript JQuery* lo que nos proporciona una gran compatibilidad con cualquier navegador web del mercado.

Al usar *hojas de estilo CSS* la apariencia de nuestra aplicación puede ser fácilmente modificada para adaptarla a nuevas tendencias de diseño de forma sencilla y eficiente.

Siempre que se termine cualquier operación, ya sea exitosa o no, el sistema nos informará a través de un sistema de notificación embebido dentro del navegador muy parecido al sistema usado en *Mac OS X* y últimas versiones de la *distribución Linux Ubuntu*.

Como mejora para un uso mucho más cómodo y rápido por parte de cualquier usuario, la aplicación dispone de un menú estático para acceder a cualquier tipo de herramienta o funcionalidad según el usuario sea un jugador de la aplicación o sea el administrador.

3.1.3. Interfaces software

El producto interactúa con el sistema operativo en el lado del servidor en dónde está instalado y con el navegador usado en el lado del cliente.

De hecho, el producto es totalmente multiplataforma tanto en el lado servidor como en el lado cliente, gracias a que el *framework RoR* está diseñado para ser instalado en entornos *Windows, Linux y Mac OS X*; mientras que en el lado cliente disponemos de varios navegadores web cualquiera que sea el sistema operativo en el que se ejecute.

Debemos destacar que el mencionado *framework* tiene una gran comunidad que lo sustenta, que da como resultado la disponibilidad de una gran cantidad de documentación y un buen soporte de la plataforma, repercutiendo directamente en la calidad del software que desarrollemos.

Hemos de indicar que *RoR* es agnóstico en cuanto al *SGBD* que utilicemos, con lo que el producto podrá ser desplegado con cualquiera de los *SGBD* más comunes como son *Oracle, MySQL, SQLite2, SQLite3 y Postgre*, a través de una *gema* específica y una simple modificación en uno de los archivos de configuración.

3.1.4. Operaciones

Existen dos tipos de usuarios dentro de la aplicación:

- **Superusuario:** Es el responsable de instalar, mantener, interconectar con la *base de datos* y optimizar el servidor dónde se encuentra la aplicación.
- **Administrador:** Es el responsable de toda la gestión de campeonatos, así como de dar de alta a los nuevos usuarios para que puedan empezar a interactuar con las funcionalidades de la aplicación.
- **Usuario:** Conforma el resto de usuarios del sistema que participan en la simulación del entorno de gestión de equipos de fútbol, a excepción de las funcionalidades exclusivas del administrador.

3.1.5. Requisitos de adaptación a la ubicación

Para poder adaptar la aplicación al entorno dónde se instalará será necesario disponer de un *SGBD* compatible con *RoR* además de modificar el fichero de configuración para conectarlo con nuestro sistema.

3.1.6. Funciones del producto

La función principal del producto que estamos presentando es la de desarrollar un videojuego multijugador dónde diversos usuarios puedan interactuar en un entorno que simule ser real a través de *Internet*. Para ello ha sido necesario crear una aplicación informática encargada de poder registrar a cualquier usuario que acceda a nuestra web y proporcionarles las herramientas necesarias para gestionar su propio club; además de simular los partidos que en una competición de fútbol se desarrollan.

Las características más importantes de esta aplicación se describen a continuación:

- Posibilidad de registro y control de acceso a una cuenta de usuario personal e intransferible que distinga al usuario unívocamente de los demás a través de una simple interfaz web.
- Capacidad para la gestión de todos los aspectos económicos de los clubes de los usuarios registrados.
- Capacidad para la visualización del calendario y clasificación de todos los clubes y usuarios registrados en el sistema.
- Capacidad para modificar la estructura deportiva y técnica de los clubes de los usuarios registrados.
- Capacidad para simular un partido de forma realista y mostrarlo a los usuarios como si se estuviera desarrollando en directo.
- Capacidad para gestionar el estado de la competición.
- Capacidad para escoger el idioma en el que se mostrará la información de nuestra aplicación.

Aquí sólo se han descrito las características generales que realiza la aplicación, se podrá comprobar toda la funcionalidad en los puntos siguientes de esta memoria, en especial en el *Manual de usuario*.

3.2. Características de los usuarios

Para el correcto manejo de la aplicación *Simulador de fútbol online* existen dos perfiles distintos de manejo.

Podríamos dividir los perfiles en:

- **Usuario (Jugador):** Necesitan unos conocimientos básicos de informática para su manejo, sobre todo orientado al manejo de aplicaciones web. También será necesario un conocimiento meramente básico de cuáles son las reglas de negocio y deportivas de un entorno de gestión de clubes deportivos de fútbol 11.

- **Usuario (Administrador):** Necesitará unos conocimientos básicos en informático para su manejo, sobre todo orientado al manejo de aplicaciones web. También será necesario conocer cómo es el discurrir de las competiciones en el diseño de nuestra aplicación.

- **Superusuario (Administrador):** Serán necesarios unos conocimientos avanzados de informática, incidiendo en conocimientos de despliegue de aplicaciones web.

3.3. Restricciones generales

El proyecto se ha desarrollado bajo la metodología *Rational Unified Process (RUP)*, proceso de desarrollo orientado a objetos, al estar basado todo el proyecto en un entorno de desarrollo bajo el paradigma de programación orientada a objetos.

El lenguaje de programación que usaremos será *Ruby*, que es libre así como el *framework* sobre el que apoyaremos nuestra aplicación web *RoR*.

No se deben tener en cuenta restricciones en el uso de memoria principal por parte de la aplicación tanto en el lado servidor como en el cliente.

Se deben tener en cuenta restricciones a la hora de crear las vistas de nuestra aplicación web para que éstas sean lo mas livianas posibles para ahorrar el mayor ancho de banda posible.

Para el desarrollo de todo el proyecto deberemos usar un *IDE* que se base en la filosofía de *software libre*, en este caso *NetBeans*.

El producto habrá de funcionar en cualquier sistema operativo que tenga disponibilidad para tener instalado un intérprete de *Ruby* en el lado servidor y en cualquier navegador del mercado compatible con los estándares web para *HTML 4.0*, *CSS 2.0*, y la librería *JQuery*. Esto hará que nuestro software tenga un mayor público objetivo de desarrolladores.

Como sistema gestor de bases de datos se podrá utilizar cualquiera que tenga una *gema* que sea capaz de conectar *RoR* con el *SGBD* escogido.

Es necesario tener en cuenta que nuestra aplicación puede correr en cualquier *pc* pero es conveniente que éste tenga la capacidad suficiente para atender varias peticiones a la vez, además de tener la configuración necesaria para que peticiones que lleguen desde *internet* puedan ser atendidas.

Para toda la creación y generación de la documentación presente hemos utilizado LaTeX.

Como controlador de la configuración hemos optado por usar *GIT*, un sistema de control de versiones distribuido que es el usado oficialmente por el proyecto *RoR* y que se integra de forma plena con *NetBeans*.

3.4. Requisitos no funcionales del sistema

3.4.1. Requisitos Generales

La solución debe cumplir como mínimo con las siguientes características basadas en las especificaciones funcionales y los requisitos no funcionales:

- Basada en la web.
- La aplicación debe esta diseñada y desarrollada sobre la plataforma *Ruby On Rails*.
- La aplicación debe ser escalable bajo estrategia vertical (Añadir más recursos al servidor) y horizontal (Añadir más servidores), según las necesidades de procesamiento.
- La aplicación debe tener bajo nivel de acoplamiento y la posibilidad de editar fácilmente los parámetros que se consideren dinámicos y requieran cambios frecuentes.
- Orientada a objetos.
- De fácil mantenimiento en cuanto a cumplimiento de estándares, uso de guías y patrones con especial énfasis en el patrón *MVC*, documentación y de fácil ubicación de componentes.
- Que permita y utilice la reutilización de código.
- Basada en la arquitectura cliente-servidor.
- La aplicación debe permitir generar avisos a través de correo electrónico.

3.4.2. Requisitos específicos

Los requisitos no funcionales generales estarán enmarcados en los siguientes aspectos:

- Escalabilidad:
  - El diseño debe contemplar el uso óptimo de recursos como las conexiones con la base de datos.
  - El diseño debe contemplar la clara división entre datos, recursos, y aplicaciones para optimizar la escalabilidad del sistema.
  - Debe contemplar requisitos de crecimiento para usuarios internos al sistema.
- Disponibilidad:
  - La disponibilidad del sistema ha de ser continua con un nivel de servicio para los usuarios de siete días y 24 horas.
  - En caso de fallos de algún subsistema no debe haber pérdida de información
  - Debe contemplar requisitos de consistencia transaccional.
- Seguridad:

    - La aplicación debe reflejar patrones de seguridad teniendo para cumplir con la ley orgánica de protección de datos.

- Mantenibilidad:
    - Se debe estructura el código de manera consistente y predecible.
    - Aprovechar al máximo las facilidades que nos aporta el *framework Ruby On Rails*, con especial énfasis en el patrón *MVC*.
- Rendimiento: La aplicación debe ofrecer una buena respuesta ante alta demanda de usuarios.

3.5. Requisitos para futuras versiones

Para futuras versiones del producto, podrían introducirse funciones avanzadas para los usuarios como selección de escudo del equipo, personalización de dorsales, avisos por sms, mercados de agentes libres, cantera y ojeadores.

Podría introducirse nuevas plantillas de diseño para personalizar la vista de diversas formas para que el usuario escoja la que más le guste.

También sería interesante establecer un apartado de estadísticas para poder llevar un seguimiento del historial en tiempo de la evolución de los equipos tanto en el plano técnico como en el económico, este apartado también sería aplicable a los futbolistas que integran los clubes.

Podría añadirse un sistema de foros y chat para una interacción entre usuarios que no se limite en exclusiva a las funcionalidades específicas de nuestro sistema.

Como restricción a futuras versiones o proyectos que estén basados en nuestra aplicación se aplicarán las referentes a las propias especificadas en el apartado de licencia de este documento.

CAPÍTULO 4

Análisis

En este capítulo vamos tratar de recopilar los requisitos funcionales, de interfaces, de rendimiento, de información y diseño de nuestro proyecto.

4.1. Metodología de desarrollo

Hemos decidido seguir una metodología de desarrollo que se adapte lo suficientemente bien al framework en torno al cual desarrollaremos nuestra aplicación.

Hemos decidido usar la metodología de desarrollo *Xtreme Programming (XP)*

La metodología será seguida dentro de unos límites y adaptada a nuestra necesidades, debido entre otras cosas a que somos nosotros mismos los que hemos decidido qué funcionalidades y qué problemas queremos resolver con nuestra aplicación; además, no hay que olvidar que no tenemos ningún equipo que coordinar ya que sólo será una persona quién se encargue de realizar todas y cada una de las funcionalidades.

La programación extrema, o *Xtreme Programming* es una de las llamadas *Metodologías Ágiles* de desarrollo de software más populares hoy en día. Inicialmente fue descrita por *Kent Beck* cuando trabajaba en la *Chrysler Corporation*. En la *XP* se da por supuesto que no se debe prever todo antes de empezar a codificar, que es imposible capturar todos los requisitos del sistema, ni saber qué es todo lo que debe hacer y por tanto es imposible hacer un diseño correcto desde el principio.

La idea principal de esta metodología consiste en trabajar estrechamente con el cliente, diseñando pequeñas versiones frecuentemente. El objetivo de estas versiones no es otro que conseguir que la aplicación funcione de la forma más simple y eficiente posible con el mínimo código. Cuando el cliente le transmite

al programador lo que necesita, éste puede hacer una estimación aproximada del tiempo que le llevará codificarlo. Cada vez que el desarrollador consigue una versión funcional de lo que el cliente ha solicitado, ésta se le muestra al cliente para que la testee y haga peticiones de las modificaciones que crea convenientes. De esta manera se evita perder tiempo desarrollando una aplicación que no sea la que el cliente esperaba. Este ciclo se repetirá tantas veces como el cliente necesite para sentirse satisfecho con la aplicación.

La *Xtreme Programming* agrupa trece prácticas básicas que se deben cumplir para asegurar el éxito del proyecto, ellas son:

- Equipo completo.
- Planificación continua.
- Test del cliente.
- Versiones pequeñas.
- Diseño simple.
- Programación por parejas.
- Desarrollo guiado por pruebas automáticas.
- Mejora continua del diseño.
- Integración continua.
- Código en multipropiedad.
- Normas de codificación comunes.
- Utilización de metáforas.
- Mantener un ritmo sostenible.

La *Xtreme Programming* consta de unos valores, unos principios fundamentales y unas prácticas. Los principios fundamentales en los que se basa son:

- Feedback: Retroalimentación veloz.
- Modificaciones incrementales.
- Refactorización
- Asunción de simplicidad.
- Respeto: Comunicación entre desarrolladores.

4.2. Descripción General

A continuación vamos a ver los factores que afectan al producto y sus requerimentos

4.3. Propósito

El propósito del siguiente apartado es definir cuáles son los requisitos que debe tener nuestro portal web que simula la gestión de un club de fútbol en un entorno multijugador y simular partidos.

Esta especificación de requisitos está destinada a ser leída por los usuarios o cualquier sujeto que tenga interés en saber cómo funciona el producto.

4.4. Ámbito

El producto que vamos a describir puede clasificarse como un videojuego multijugador online integrado en un servidor web en la cual los usuarios del portal se dedicarán a gestionar un club de fútbol en la parcela económica y deportiva, y dispondrá de un simulador de los partidos entre clubes.

El producto ha de ofrecer una interfaz lo bastante sencilla para que cualquier usuario pueda gestionar las distintas facetas en escasos minutos. Dado que la aplicación es de tipo web, deberemos de tener especial cuidado en respetar los estándares para que la aplicación sea visible en cualquier navegador del mercado obteniendo así un videojuego que sea totalmente multiplataforma. La tecnología que usaremos en el lado del servidor web será el framework Ruby On Rails.

4.5. Perspectiva del Producto

El presente producto ha de correr en un servidor que sea compatible con *Ruby on Rails*. Antes que nada *Ruby On Rails* es un framework que hace mas fácil desarrollar, implementar y mantener aplicaciones web. Esto es debido entre otras razones a que:

- Toda aplicación web implementada usa el esquema ***MVC*** (Modelo Vista Controlador).
- Toda pieza de código tiene su lugar y todas ellas interactúan según un camino estándar, es lo que ha venido a llamarse ***COC*** (Convención sobre Configuración).
- Las aplicaciones en Rails están escritas en Ruby, un lenguaje de script moderno y orientado a objetos que ayuda a que se potencie otro de los pilares de la filosofía de desarrollo que viene abreviado en la siguiente nomenclatura: ***DRY*** (No te repitas).

En cuanto a la máquina necesaria para ejecutar el servidor, valdrá cualquier sistema que sea compatible con *Windows*, *Linux* o *MacOS X* recomendándose que se use un hardware acorde con las peticiones que el portal reciba. Los usuarios deberán de acceder al portal mediante cualquier navegador web disponible en el mercado que sea compatible con los estándares web ***XHTML 1.1 Strict*** y ***CSS 2.0.***

4.6. Características del usuario

A continuación vamos a ver a qué tipo de usuarios está dirigido el producto y cómo afectan éstos a las funciones que debe realizar nuestro producto.

El producto está dirigido a todo usuario que sepa manejar un navegador web y que esté interesado en la simulación de la gestión de un club de fútbol, así como conocer las reglas y premisas básicas de negocio del deporte.

Identificamos dos roles bien diferenciados en nuestro sistema:

1. **Usuario Jugador**: Se refiere a una persona que se registra en el portal web y que participará en la gestión de un club al que se le asignará. Este usuario podrá conectarse al sistema e interactuar con el mismo a través de la interfaz web.
2. **Usuario Administrador**: Es un tipo de usuario que será capaz de modificar el comportamiento del sistema a través del menú de administración de nuestra aplicación para llevar a cabo las acciones de gestión de la competición de nuestro sistema.

4.7. Obligaciones generales

Uno de los aspectos que influirá determinantemente en el éxito del producto será la eficiencia en las peticiones y el diseño de una intertaz amigable y sencilla para poder gestionar los diferentes parámetros del club de una forma rápida y precisa.

De cara al diseño de la interfaz web, uno de los mayores desafíos con los que nos encontraremos será el conseguir que el diseño de la interfaz sea operativa en cualquiera de los navegadores del mercado, ya que los dos navegadores más populares no soportan actualmente con suficiente solvencia los estándares ***CSS 2.0*** (*Internet Explorer 8* y *Firefox 3.5*).

4.8. Suposiciones y Dependencias

El presente producto depende enteramente de que el usuario disponga de una conexión red al servidor a través de un navegador web (probablemente ***internet***), por tanto es importante respetar los estándares pertinente que estipula la ***w3c***.

4.9. Especificación de los requisitos del sistema

En esta sección trataremos de especificar todos los requisitos de nuestro sistema software y conforme a ello elaboraremos un documento de especificación de requerimentos.

4.9.1. Requisitos de interfaces externas

Tenemos que describir de forma detallada los requisitos de conexión a otros sistemas hardware o software con los que vamos a interactuar.

4.9.1.1. Requisitos de interfaces de usuario

Podemos dividir los requisitos de interfaz de usuario según tres roles:

- **Usuario Jugador y Usuario Administrador:**
  - Interfaz atractiva para los usuarios: Es fundamental para la buena aceptación de nuestra aplicación el crear una interfaz lo suficientemente clara, dinámica y estética para que el usuario sienta una inmersión en la experiencia de juego. Una interfaz no atractiva sólo traerá problemas de rechazo y disconformidad por parte del usuario.

    Para que la interfaz sea sencilla de modificar y tratándose de una aplicación web usaremos *hojas de estilo CSS*, así todo el apartado de diseño gráfico podrá ser modificado posteriormente sin que por ello afecte al contenido ni la estructura del documento *HTML*.
  - Una característica de nuestra aplicación es que el sistema de avisos está embebido dentro de la página web y es muy parecido al sistema de notificación de los sistemas operativos *Mac OS X* y de la *distribución Linux Ubuntu*.
- **Superusuario:** En este caso la interfaz corresponderá a la que tenga el propio sistema operativo dónde esté instalado la aplicación.

4.9.1.2. Interfaces con el hardware

Las copias de seguridad del sistema se realizarán de forma periódica cada día, semana y mes de forma incremental y utilizando la herramienta *rsync*. Dichas copias de seguridad se almacenarán en otro servidor dedicado exclusivamente al almacenaje de las copias.

4.9.1.3. Interfaces con el software

La aplicación web correrá en el lado del servidor en un sistema *Linux* con *Ubuntu Server 12.04*. Mientras que en el lado cliente podrá ejecutarse en cualquier sistema que disponga de un navegador web que cumpla los estándares 2.0 *W3C*.

4.9.2. Requisitos de rendimiento

Este aspecto es fundamental en el análisis del sistema, se ha de tener en cuenta que la aplicación es de tipo servidor y que a ella estarán conectados concurrentemente un número de usuarios elevado.

Se ha considerado también el rendimiento que nos ofrece el diseño web realizado. Para que nuestra página web sea compatible con la mayoría de dispositivos y se ejecute lo suficientemente fluida hemos decidido no usar la tecnología propietaria *Flash*, usando para la renderización de efectos la librería *JQuery* en combinación con hojas de estilo *CSS*.

El rendimiento de la aplicación podrá ser menor en la primera instanciación de la aplicación debido a que no se habrá cacheado ningún contenido.

4.9.3. Requisitos de Información

La *base de datos* es uno de los componentes fundamentales de la aplicación. En ella se van a almacenar todos los datos registrados por la aplicación.

La aplicación sólo hará uso de una conexión al *SGBD* escogido por el *superusuario* en la configuración de instalación de la aplicación, cuyo nombre también podrá ser escogido por el citado *superusuario*.

El sistema almacenará en forma de tablas la siguiente información relativa a las siguientes entidades:

User Se almacenan los datos de cada uno de los usuarios que interactúan con la aplicación.

Club Se almacenan los datos de los clubes que participan en las ligas.

League Se almacenan los datos de las ligas que existen en el sistema.

Season Se almacenan los datos de las temporadas correspondientes a cada liga del sistema.

Round Se almacenan los datos de las jornadas de cada temporada.

MatchGeneral Se almacenan los partidos que se han de jugar cada jornada.

MatchDetail Se almacenan las acciones que tienen lugar en cada partido.

LineUp Se almacenan las alineaciones de cada equipo y partido.

ClubFinancesRound Se almacenan los datos de finanzas correspondientes a cada jornada y club.

Player Se almacenan los datos de los jugadores de fútbol de los clubes.

Offer Se almacenan los datos de las ofertas que se realizan los usuarios de la aplicación sobre los jugadores de fútbol.

Training Se almacenan los datos de los entrenamientos de los jugadores de fútbol.

AdminMessages Se almacenan los mensajes de administración hacia los usuarios.

Toda la información sobre la *base de datos* será ampliada en su sección correspondiente.

La tabla *User* contiene los datos de los usuarios que interactúan con la aplicación, en ella se almacenará la contraseña de cada uno de ellos; es por ello que será necesario un *algoritmo criptográfico* denominado

MD5. De esta forma conseguimos proteger la contraseña de los usuarios frente a un posible ataque al servidor dónde esté la *base de datos*.

Es necesario que el *superusuario* conozca el *SGBD* que será utilizado con nuestra aplicación, además de disponer de un usuario y contraseña para la correcta creación de las tablas.

4.9.4. Restricciones de diseño

No existe ninguna restricción considerable en el diseño del sistema.

Todo el diseño del sistema se debe de regir con la especificación realizada.

Es importante estimar cuál va ser el tráfico que nuestra aplicación va sufrir para disponer de un *servidor* y una conexión lo suficientemente potentes para dar servicio a todos los usuarios que se conecten a la aplicación.

4.9.5. Atributos del sistema software

El sistema ha de cumplir las siguientes propiedades:

- **Fiabilidad**: Este sistema, como todo software de calidad, debe ser fiable. Al estar tratando de una aplicación web que ha de tener un elevado acceso concurrente a los datos, hemos de cerciorarnos que las operaciones no dejen el sistema en un estado inconsistente por una posible caída del servidor.

  Por ello el sistema se encarga de generar un *log* con todas las operaciones que ocurren en nuestra aplicación.

  Hemos de tener en cuenta que el usuario de nuestra aplicación no ha de conocer nada sobre la estructura interna de nuestra aplicación, por ello los mensajes de error serán lo más simples posibles para evitar posibles ataques al servidor dónde es alojada.

- **Disponibilidad**: La disponibilidad de este producto debe ser plena. La aplicación ha de estar disponible en cualquier momento y ser accesible desde cualquier lugar. Para llevar a cabo este precepto será necesario alojar la aplicación en un lugar web que tenga la suficiente capacidad para dar servicio a un elevado número de usuarios.

- **Seguridad**: La seguridad de este tipo de aplicaciones ha de ser bastante elevada para conseguir cumplir leyes de protección de datos y asegurar un correcto funcionamiento.

  Para la seguridad y confidencialidad de los datos se debe seguir un protocolo dirigido a asegurar la confidencialidad de los datos almacenados en la *base de datos*.

  Será necesario que el superusuario de la aplicación disponga de un nombre de usuario y contraseña lo suficientemente seguros para trabajar con el *SGBD* escogido.

  Los usuarios de la aplicación tendrán un nombre usuario y una contraseña asociada que encriptada con el algoritmo *MD5*.

Es posible que cualquier persona acceda a la página web para estar informado de los que ocurre en las diferentes ligas que componen el juego. Este tipo de usuarios no necesitan disponer de nombre de usuario y contraseña asociada.

- **Mantenibilidad**: La mantenibilidad de la aplicación es un aspecto clave del desarrollo de aplicaciones web debido al constante avance que sufre este tipo de tecnologías. Gracias a desarrollar la aplicación en base a un *framework de desarrollo rápido de aplicaciones* conseguimos una alta mantenibilidad y legibilidad del *código fuente* de la aplicación.
- **Portabilidad**: La aplicación web, al estar desarrollada bajo *Ruby On Rails* tiene una amplia portabilidad en cuánto al lado servidor ya que éste es compatible con los principales sistemas operativos del mercado.

  En cuanto al lado cliente tenemos la portabilidad asegurada ya que sólo será necesario que el usuario disponga de una conexión a *internet* y un navegador web compatible con los estándares.

4.10. Descripción de Requisitos Funcionales

Configurar el sitio

El administrador podrá navegar por la página de configuración web y la usará para cambiar el comportamiento de la misma.

Esta función se realizará raras veces, entendiendo por estas, las ocasiones en las que el administrador, quiera ejecutar la simulación de los partidos, el pasar a la siguiente jornada, la creación de nuevas ligas y la promoción de clubes.

Crear una cuenta de usuario

Para que un usuario cualquiera sea capaz de poder usar nuestro producto ha de tener una cuenta registrada en el servidor que lo distinga unívocamente de cualquier otro usuario; para ello se le pedirá introducir un nombre de usuario, y una dirección de e-mail. Una vez registrado el sistema enviará un e-mail de confirmación de registro dando la bienvenida a nuestro sistema.

Esta función es condición necesaria para que puedan realizarse todos y cada uno de los requisitos que se expresan en los apartados siguientes y se realizará una única vez por usuario.

Acceder a la cuenta de usuario

Para que cualquier usuario registrado sea capaz de usar nuestro producto deberemos de proporcionarle un método de acceso e identificación mediante el nombre de cuenta y contraseña que han sido albergados en nuestro servidor.

Esta operación se realizará muy a menudo y tantas veces como el usuario desee teniendo como resultado que el usuario estará logueado en nuestro servidor.

Abandonar la cuenta de usuario

Cualquier usuario que haya iniciado sesión en nuestra aplicación deberá ser capaz de abandonar de forma segura la misma.

Esta operación se realizará muy a menudo y tantas veces como un usuario que haya iniciado sesión desee dando como resultado que el usuario deja de estar logueado en nuestro servidor.

Abrir equipos

El administrador será capaz de decidir cuándo los usuarios del sistema podrán modificar sus alineaciones o tácticas y entrenamientos, así como la posibilidad de interactuar con los sistemas de finanzas y ofertas de jugadores.

Esta función será realizada muy a menudo debido a que se habrá de cambiar de jornada de forma periódica para simular el curso normal del calendario de un campeonato liguero. El resultado de esta acción será la posibilidad de modificar alineaciones, tácticas, entrenamientos y realizar ofertas por futbolistas entre usuarios.

Cerrar equipos

El administrador será capaz de decidir cuándo los usuarios del sistema no podrán modificar sus alineaciones, tácticas y entrenamientos, así como la posibilidad de interactuar con los sistemas de finanzas y ofertas de jugadores.

Esta función será realizada muy a menudo debido a que se habrá de cambiar de jornada de forma periódica para simular el curso normal del calendario de un campeonato liguero. El resultado de esta acción será la posibilidad de modificar alineaciones, tácticas, entrenamientos y realizar ofertas por futbolistas entre usuarios y será posible por parte del administrador el simular y avanzar en el calendario liguero.

Listar ofertas

El usuario podrá en cualquier momento visionar el listado de ofertas pendientes, aceptadas o rechazadas que tenga en su historial.

Esta operación podrá ser realizada en cualquier momento y llevará como resultado que el usuario será capaz de ver el historia de ofertas por los futbolistas de su club o las emitidas por él mismo.

Emitir oferta por futbolista

El usuario podrá en cualquier momento hacer una oferta por un futbolista en el que esté interesado, interactuando de este modo con otros usuarios a través de acuerdos económicos. Para evitar que dos usuarios

puedan realizar acuerdos económicos para hacer trampas y beneficiarse deportiva y económicamente, éstos no podrán tasar a sus propios jugadores haciendo de la oferta una especie de pujo, sino que será el propio sistema el que estipulará el precio del futbolista en base a la calidad que éste disponga.

Esta operación podrá ser realizada en cualquier momento en el que los equipos estén abiertos y tendrá como resultado la emisión de una oferta económica a un jugador de un club ajeno que será recibida por el usuario que lo posee.

Aceptar oferta por futbolista

El usuario podrá en cualquier momento aceptar una oferta de un futbolista de su propiedad emitida por otro usuario.

Esta operación podrá ser realizada en cualquier momento en el que los equipos estén abiertos y tendrá como resultado que la transacción del jugador entre los clubes sea llevada a cabo con la consecuente actualización de la economía y plantillas de ambos clubes.

Rechazar oferta por futbolista

El usuario podrá en cualquier momento rechazar una oferta de un futbolista de su propiedad emitida por otro usuario.

Esta operación podrá ser realizada en cualquier momento en el que los equipos estén abiertos y tendrá como resultado que el club que emitió la oferta tendrá una confirmación de rechazo por la misma.

Cancelar oferta por futbolista

Un usuario que esté arrepentido de una oferta realizada por un futbolista podrá cancelarla en cualquier momento.

Esta operación podrá ser realizada en cualquier momento en el que los equipos estén abiertos y tendrá como resultado que la oferta que se emitió por un futbolista en concreto será revocada.

Renovación de futbolistas

El sistema actualizará los sueldos que reciben los futbolistas al término de cada temporada, según la ponderación de los atributos cualitativos que tengan.

Esta operación se realizará una vez por temporada y afectará a todos los futbolistas del sistema con ello se actualizarán los pagos y clausulas de los jugadores.

Venta de entradas

El usuario podrá estipular el precio de venta de las entradas en los partidos que se disputen en su estadio siendo ésta la forma principal de obtención de ingresos para la gestión económica del club.

Esta función será realizada tantas veces como desee el usuario, siempre y cuando los equipos estén abiertos y tendrá como resultado el cambio en el precio de los tickets, lo cual afectará al comportamiento de asistencia al estadio y la recaudación por los partidos jugados en casa.

Planificación de alineaciones

El usuario podrá escoger cuál será la alineación que presentará para el próximo partido que vaya a disputar su equipo, pasando ésta a ser la alineación por defecto hasta que el usuario decida volver a cambiarla.

Esta operación se realizará a menudo, siempre y cuando los equipos estén abiertos y tendrá como resultado que los jugadores puedan ser alineados de la forma que el usuario crea conveniente para afrontar el siguiente partido.

Planificación de tácticas

El usuario podrá escoger entre diversas tácticas de juego para afrontar los distintos partidos. Esta acción repercutirá directamente en el estilo de juego y por tanto en la simulación resultado del partido que dispute.

Esta operación se realizará tantas veces como el usuario desee, siempre y cuando los equipos estén abiertos y tendrá como resultado que los jugadores se dispongan de una forma diferente en el terreno de juego, afectando de forma directa a la ponderación de los atributos para el cálculo de calidades.

Planificación de entrenamientos

El usuario podrá personalizar los atributos a entrenar de los futbolistas integrantes de su equipo. De esta forma el club adquirirá valor deportivo y económico.

Esta función se realizará a menudo y siempre y cuando los equipos estén abiertos, teniendo como resultado que se incremente de forma consistente el atributo que el usuario ha decidido mejorar en el jugador.

Asignar club

Una vez que un usuario haya sido registrado en el servidor mediante una cuenta de usuario el sistema le asignará un nuevo club cuando todas las ligas hayan concluido y el administrador así desee.

Esta operación se realizará una vez por temporada y por cada nuevo usuario registrado en el sistema con los equipos cerrados, siempre y cuando que así lo desee el administrador, teniendo como objetivo el crear nuevas ligas que disputen una competición jerárquica en la que el usuario tenga el reto de ir ascendiendo.

Simular jornadas

El sistema debe simular la disputa de un partido en base a la ponderación de los atributos cualitativos de los futbolistas alineados, tácticas empleadas y variables aleatorias confrontadas con las del equipo rival.

Esta función se realizará a menudo, una vez por jornada y siempre y cuando los equipos estén cerrados. De esta forma generamos todos los comentarios y acciones que se llevarán a cabo en el partido disputado de cada jornada.

Comenzar jornadas

El sistema debe emitir comentarios durante el transcurso del partido para que cualquier usuario sea capaz de conocer la evolución del mismo.

Esta función se realizará a menudo, una vez por jornada y con los equipos cerrados. De esta forma se irán comentando los partidos de forma que parezcan que se están simulando en tiempo real.

Pasar a siguiente jornada

El sistema debe ser capaz de gestionar el calendario creado de cada competición. Para ello se le da la oportunidad al administrador de avanzar en la línea temporal del calendario para hacer discurrir las jornadas del calendario.

Además, en esta acción se actualizarán los entrenamientos y traspasos aceptados por futbolistas entre usuarios, así como restar los gastos de la plantilla y mantenimiento del estadio.

Esta función se realizará a menudo a lo largo de la competición y con los equipos cerrados. De esta forma se actualizan los traspasos, calidades de jugadores y las clasificaciones de las ligas de forma ordenada y coherente.

Promoción y descenso de clubes

El sistema debe ser capaz de gestionar varias ligas organizadas de forma jerárquica según el número de usuarios que estén registrados en nuestro servidor. De esta forma conseguiremos que el usuario desee mejorar la posición de su club, lo que repercutirá directamente en la capacidad deportiva y económica del mismo.

Esta función se realizará una vez por temporada y una vez por jornada, teniendo que estar los equipos cerrados. Una vez realizada la acción los tres últimos equipos de una liga descenderán a la inmediatamente inferior, y los tres primeros ascenderán a la inmediatamente inferior.

Composición de calendarios

El sistema deberá de crear un calendario de partidos por cada liga que está albergada en el servidor. Los calendarios se crearán una vez por temporada y será accesible a todos los usuarios.

Esta función se realizará una vez por creación de liga, con los equipos cerrados y tendrá como objetivo el crear un calendario de jornadas por cada liga en la que se enfrentarán todos los equipos contra todos los equipos dos veces, una como visitante y otra como local.

Gestiones de lesiones y sanciones

El sistema debe ser capaz de llevar a cabo una política realista de imposición de sanciones a los futbolistas, así como simular lesiones más o menos duraderas.

Esta función se realizará a menudo.

4.11. Especificación de los Casos de Uso

Caso de uso crear cuenta de usuario

Descripción: Crea una cuenta de usuario para poder participar y acceder a las acciones provistas por nuestra aplicación.
Actores: Usuario no registrado, usuario registrado, sistema.
Precondición: Ninguna.
Postcondición: Se da de alta un nuevo usuario en el sistema.
Escenario principal:

1. Un usuario no registrado desea registrarse.
2. El usuario introduce el nombre de usuario, la contraseña asociada al mismo, una cuenta de correo electrónico además de repetir la contraseña para evitar un error humano al ingresarla y también el idioma en el que desea ver por defecto la web y recibir los correos electrónicos.
3. El sistema registra al usuario.
4. El sistema avisa al usuario registrado mediante un correo electrónico y la propia interfaz web que ha completado su registro con éxito.

Escenario alternativo:

1. Si el usuario introduce un nombre de cuenta de usuario o e-mail existente o dos contraseñas no coincidentes se le avisará de los errores cometidos para que pueda realizar un registro exitoso.

Caso de uso acceder a la cuenta de usuario

Descripción: El usuario provee un sistema para poder identificar a los usuarios registrados en el mismo y dejarlos acceder a la funciones principales de nuestro sistema.
Actores: Usuario registrado, usuario logueado.
Precondición: El usuario ha de estar registrado en nuestro sistema.
Postcondición: Se identifica al usuario en el sistema y se le permite acceder a las funciones principales del mismo.
Escenario principal:

1. Un usuario registrado desea iniciar sesión.
2. El usuario introduce el nombre de usuario y la contraseña asociada al mismo, además de repetir la contraseña para evitar un error humano al ingresarla.
3. El sistema registra al usuario.
4. El sistema avisa al usuario registrado mediante un correo electrónico y la propia interfaz web que ha completado su registro con éxito.

Escenario alternativo:

1. Si el usuario introduce un nombre de cuenta de usuario o dos contraseñas no coincidentes se le avisará de los errores cometidos para que pueda realizar un registro exitoso.

Caso de uso abandonar la cuenta de usuario

Descripción: El sistema provee un sistema para que un usuario logueado pueda desconectarse de forma segura de nuestro sistema.
Actores: Usuario logueado.
Precondición: El usuario ha de haber iniciado sesión en nuestro sistema.
Postcondición: Se cierra la sesión del usuario con nuestro sistema.
Escenario principal:

1. El usuario logueado desea abandonar su sesión.
2. Se le pregunta al usuario si está seguro de realizar la acción.
3. El sistema avisa al usuario que acaba de finalizar la sesión a través de la interfaz web de nuestra aplicación.

Escenario alternativo:

1. Si el usuario decide que no está seguro de abandonar la sesión no se hace nada.

Caso de uso abrir equipos

Descripción: El sistema provee un sistema para que el usuario administrador pueda permitir cambios de jugadores, tácticas o cualquier tipo de acción que modifique el estado de los equipos.
Actores: Usuario administrador.
Precondición: Ninguna.
Postcondición: Los equipos quedarán abiertos.
Escenario principal:

1. El administrador cierra los equipos.
2. El sistema avisa al administrador que acaba de abrir los equipos a través de la interfaz web de nuestra aplicación.

4.11.1. Caso de uso cerrar equipos

Descripción: El sistema provee un sistema para que el usuario administrador pueda evitar cambios de jugadores, tácticas o cualquier tipo de acción que modifique el estado de los equipos.
Actores: Usuario administrador.
Precondición: Ninguna.
Postcondición: Los equipos quedarán cerrados.
Escenario principal:

1. El administrador cierra los equipos.
2. El sistema avisa al administrador que acaba de cerrar los equipos a través de la interfaz web de nuestra aplicación.

Caso de uso de notificación de registro por e-mail

Descripción: El sistema notificará por correo electrónico que un registro asociado al e-mail proporcionado se ha efectuado correctamente.
Actores: Sistema y usuario registrado
Precondición: El usuario ha de haber realizado un registro exitoso.
Postcondición: El sistema remite un correo electrónico en el idioma que seleccionó en el registro nuestro usuario.
Escenario principal:

1. El sistema detecta que un nuevo usuario ha sido creado.

2. Remite un correo electrónico de bienvenida a la aplicación a la cuenta de correo asociada al usuario registrado.

Caso de uso de listar ofertas emitidas

Descripción: El sistema ha de proporcionará una interfaz para poder listar el historial de ofertas emitidas del club asociado.
Actores: Usuario logueado.
Precondición: El usuario ha de estar logueado en nuestra aplicación.
Postcondición: Se muestra un listado del historial de ofertas emitidas del club.
Escenario principal:

1. Se muestra un listado del historial de ofertas emitidas del club.

Caso de uso de listar ofertas recibidas

Descripción: El sistema ha de proporcionará una interfaz para poder listar el historial de ofertas recibidas del club asociado.
Actores: Usuario logueado.
Precondición: El usuario ha de estar logueado en nuestra aplicación.
Postcondición: Se muestra un listado del historial de ofertas recibidas del club.
Escenario principal:

1. Se muestra un listado del historial de ofertas recibidas del club.

Caso de uso de emitir oferta por futbolista

Descripción: El sistema ha de proporcionará una interfaz para poder realizar una oferta por un futbolista ajeno al club del que el usuario es propietario.
Actores: Usuario logueado.
Precondición: El usuario ha de estar logueado en nuestra aplicación, el comprador ha de tener la suficiencia económica
para acometer el fichaje, además los equipos han de estar abiertos.
Postcondición: El usuario emite una oferta por un futbolista ajeno a su club y el propietario del mismo recibe la misma.
Escenario principal:

1. Un usuario emite una oferta por un futbolista que posee otro usuario.
2. El usuario propietario del futbolista recibe la oferta.

Escenario alternativo:

1. El usuario comprador no puede efectuar la oferta.

2. El sistema emitirá un aviso a través de la interfaz web para informar que es imposible emitir la oferta.

Caso de uso de cancelar oferta emitida por futbolista

Descripción: El sistema ha de proporcionará una interfaz para poder cancelar una oferta emitida por un futbolista.
Actores: Usuario logueado.
Precondición: El usuario ha de estar logueado en nuestra aplicación, la oferta ha de existir y los equipos han de estar abiertos.
Postcondición: El usuario cancela una oferta por un futbolista de un equipo ajeno al suyo. **Escenario principal:**

1. El usuario cancela la oferta previamente realizada.

Caso de uso de rechazar oferta por futbolista

Descripción: El sistema ha de proporcionará una interfaz para poder rechazar una oferta realizada por un futbolista que provenga de otro usuario. **Actores:** Usuario logueado.
Precondición: El usuario ha de estar logueado en nuestra aplicación, la oferta ha de existir y los equipos han de estar abiertos.
Postcondición: El usuario rechaza una oferta por un futbolista de un equipo ajeno al suyo.
Escenario principal:

1. El usuario rechaza la oferta previamente realizada.

Caso de uso de aceptar oferta por futbolista

Descripción: El sistema ha de proporcionará una interfaz para poder aceptar una oferta realizada por un futbolista que provenga de otro usuario.
Actores: Usuario logueado.
Precondición: El usuario ha de estar logueado en nuestra aplicación, la oferta ha de existir, los equipos han de estar abiertos, la venta del jugador no debe suponer rebasar el límite mínimo de jugadores en plantilla.
Postcondición: El usuario acepta la oferta, el montante económico de la clausula del jugador es transferido a las arcas del club y el jugador ya no es perteneciente a la plantilla.
Escenario principal:

1. El usuario acepta la oferta.

2. El futbolista es transferido a la plantilla del usuario emisor de la oferta.
3. Se incrementa la caja del club en el montante económico de la clausula del jugador transferido.

Primer Escenario alternativo:

1. El usuario emisor no tiene recursos económicos suficientes para acometer el fichaje.
2. El sistema informará al usuario que no puede aceptar la oferta y ésta será automáticamente rechazada.

Segundo Escenario alternativo:

1. El usuario receptor de la oferta está en el número mínimo de jugadores en plantilla.
2. El sistema informará al usuario que no puede aceptar la oferta y la misma será automáticamente rechazada.

Caso de uso de renovación de futbolistas

Descripción: El sistema renovará cada final de temporada a los jugadores de nuestro equipo actualizando el sueldo que reciben y su clausula de rescisión.
Actores: Sistema.
Precondición: Las temporadas han de haber concluido y los equipos bloqueados.
Postcondición: Los jugadores serán renovados en función de la nueva ponderación de sus atributos así como su clausula de rescisión.
Escenario principal:

1. El sistema realizará una ponderación de los atributos de los futbolistas.
2. Se asigna una clausula de rescisión en función de la ponderación.
3. Se asigna un sueldo anual en función de la ponderación.

Caso de uso de fijar precio de venta de entradas

Descripción: El usuario podrá modificar el precio de las entradas de su equipo de fútbol. Este hecho repercutirá directamente en la afluencia al estadio por parte de nuestros aficionados y a las arcas del club.
Actores: Usuario logueado.
Precondición: Los equipos deben estar desbloqueados
Postcondición: El precio del ticket de entrada es modificado.
Escenario principal:

1. El usuario desea cambiar el precio del ticket de entrada.

2. El ticket de entrada es cambiado al precio estipulado por el usuario.

Escenario alternativo:

1. El usuario introduce un precio menor de cero.
2. El sistema informará al usuario que no puede asignar un precio negativo a través de la interfaz web.

Caso de uso de planificación de alineaciones

Descripción: El usuario podrá modificar quiénes serán los futbolistas que jueguen el siguiente partido.
Actores: Usuario logueado
Precondición: Los equipos deben estar desbloqueados.
Postcondición: El usuario dejará formada la alineación que usará en el siguiente partido.
Escenario principal:

1. El usuario desea cambiar la posición entre dos jugadores.
2. El usuario seleccionará dos jugadores.
3. Efectuará el cambio de posición.
4. El usuario repetirá este proceso todas las veces que necesite hasta dejar la alineación como desee.

Escenario alternativo:

1. El usuario selecciona un sólo jugador.
2. Efectúa el cambio de posición.
3. El sistema emitirá un aviso para informar que la operación que intenta hacer es imposible.

Caso de uso de planificación de tácticas

Descripción: El usuario podrá seleccionar una táctica distinta para afrontar el próximo partido.
Actores: Usuario logueado.
Precondición: Los equipos han de estar desbloqueados.
Postcondición: Se cambia la táctica por la seleccionada por el usuario.
Escenario principal:

1. El usuario desea cambiar la táctica usada en el partido.
2. El usuario selecciona la táctica que desea usar y la fija como táctica por defecto para los siguientes partidos.

4.11.2. Caso de uso de planificación de entrenamientos

Descripción: El usuario podrá mejorar una de las habilidades que integran un jugador cada vez que lo mande a entrenar.
Actores: Usuario logueado.
Precondición: Los equipos han de estar desbloqueados y el jugador no puede estar entrenando otra habilidad previa.
Postcondición: El jugador entrenará la habilidad seleccionada por el usuario las jornadas que el sistema estipule.
Escenario principal:

1. El usuario desea que un jugador de su plantilla entrene cierta habilidad.
2. El sistema informará de cuántas jornadas serán necesarias para terminar el entrenamiento.

Escenario alternativo:

1. Si el usuario intenta poner a entrenar la habilidad de un jugador que haya llegado al máximo posible para el atributo el sistema le informará a través de la interfaz web de que no es necesario realizar dicha acción.

Caso de uso asignar club

Descripción: El sistema proporciona una forma de asignar nuevos club de fútbol a los nuevos usuarios registrados cuando el administrador lo desee entre temporada y temporada.
Actores: Sistema, usuario registrado y administrador
Precondición: Las temporadas han de haber finalizado, debe haber un mínimo de nuevos usuarios registrados para poder formar la nueva liga y los equipos han de estar bloqueados.
Postcondición: Se les asigna a cada usuario un nuevo club con varios jugadores y un monto inicial de dinero para poder comenzar con las transacciones.
Escenario principal:

1. El administrador desea crear una nueva liga.
2. El sistema verifica que haya los suficientes nuevos usuarios sin clubes asignados para crearla.
3. El sistema genera una plantilla de futbolistas nuevos a cada nuevo club creado.
4. Se asigna cada club a cada usuario nuevo que formará la nueva liga.

Caso de uso de simular jornadas

Descripción: El sistema simulará cada partido de cada jornada de cada temporada según las alineaciones y tácticas de los equipos que se enfrenten.

Actores: Administrador y Sistema.
Precondición: Los equipos deben estar bloqueados.
Postcondición: Se generará un conjunto de comentarios ordenados por cada acción que tenga lugar en el partido.
Escenario principal:

1. El administrador decidirá simular los partidos de la jornada en la que se encuentre el sistema.
2. El sistema generará una ristra de comentarios por cada acción generada en el encuentro usando la ponderación de los equipos alineados y las tácticas utilizadas.

Caso de uso de comenzar jornadas

Descripción: El sistema emulará la retransmisión de los partidos englobados en las jornadas en tiempo real.
Actores: Administrador y Sistema.
Precondición: Los equipos deben estar bloqueados.
Postcondición: Se emitirá a través de la interfaz web el conjunto de comentarios generados previamente en la simulación de los partidos.
Escenario principal:

1. El administrador decide que se comiencen a emitir los comentarios de los partidos.
2. El sistema empieza a mostrar los comentarios de cada partido de forma ordenada y actualizándose cada minuto.

Caso de uso de pasar a siguiente jornada

Descripción: El sistema proveerá una forma de avanzar a lo largo del calendario de partidos y actualizar traspasos, finanzas y entrenamientos.
Actores: Administrador y Sistema.
Precondición: Los equipos deben estar bloqueados
Postcondición: Se pasará de jornada en el calendario de las ligas, se actualizarán los entrenamientos, traspasos y finanzas de los clubes registrados en el sistema.
Escenario principal:

1. El administrador decide pasar a la siguiente jornada.
2. El sistema actualiza los resultados y las clasificaciones de cada liga y hace de la jornada siguiente la actual.
3. Se actualizan las finanzas de los clubes.
4. Se actualizan los traspasos aceptados.
5. Se actualizan los entrenamientos de los jugadores.

Escenario alternativo:

1. Si la jornada es la última se dará por concluidas las temporadas que tenían lugar en cada liga.

Caso de uso de promoción y descenso de clubes

Descripción: El sistema proveerá una forma de organizar las ligas de forma jerárquica.
Actores: Administrador y Sistema.
Precondición: Los equipos deben estar bloqueados
Postcondición: Se ascenderán y descenderán los clubes que estén en posiciones de descenso y ascenso según la organización jerárquica de las ligas.
Escenario principal:

1. El sistema reorganizará las ligas para reflejar los cambios en las ligas debida a las posiciones que obtengan en la tabla de clasificación los equipos de nuestro sistema.

Caso de uso de composición de calendarios

Descripción: El sistema proveerá una forma de componer los calendarios de las ligas que se estén jugando en nuestra aplicación.
Actores: Administrador y Sistema.
Precondición: Los equipos deben estar bloqueados y las temporadas finalizadas.
Postcondición: Se generará un calendario de jornadas siguiendo un algoritmo round-robin de creación de calendarios por cada liga albergada en nuestro sistema.
Escenario principal:

1. El sistema creará un calendario por liga de las nuevas temporadas que van a jugarse en nuestro sistema.

4.12. Diagramas de Casos de Uso

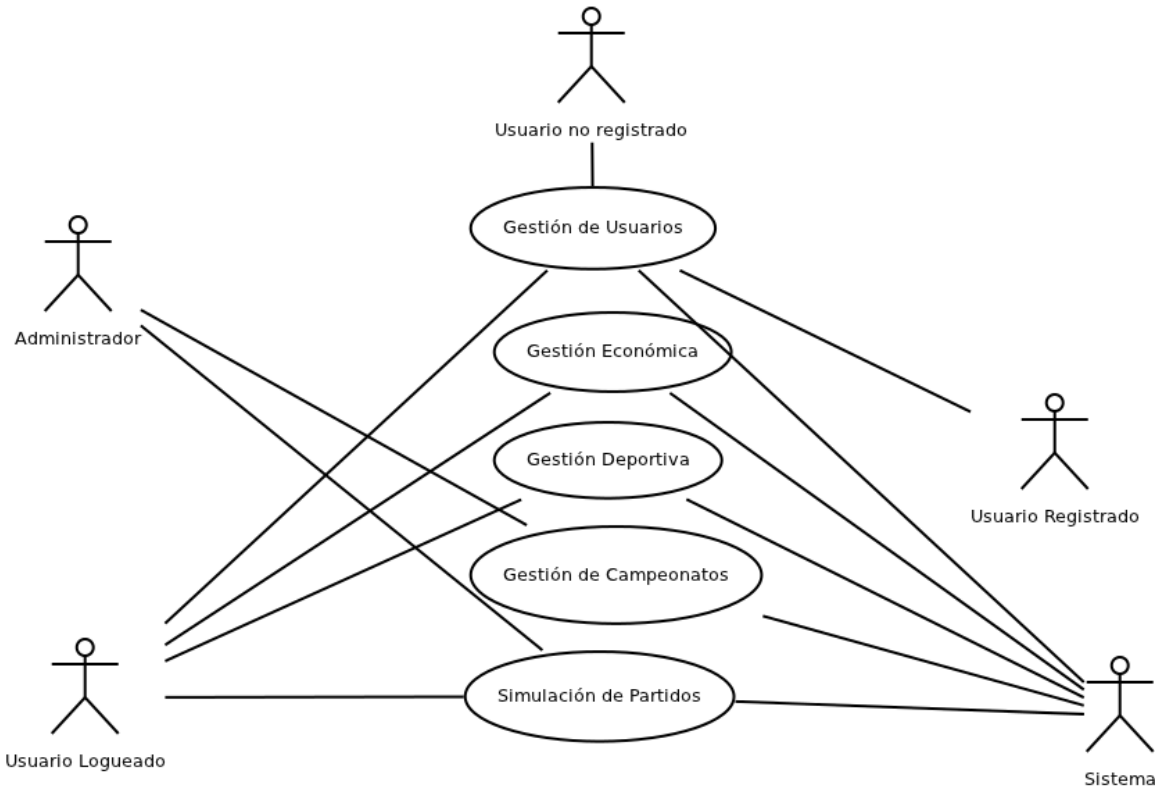

Figura 4.1: Diagrama de casos de uso de las funciones principales

Descripción:

1. Gestión de usuarios

    - Configurar el sitio
    - Crear una cuenta de usuario
    - Acceder a la cuenta de usuario
    - Abandonar la cuenta de usuario
    - Notificación de registro por e-mail

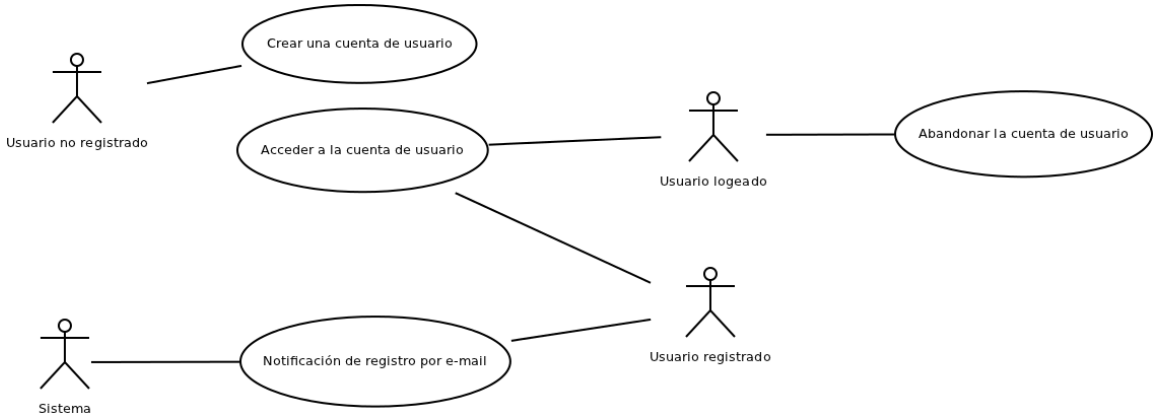

Figura 4.2: Diagrama de casos de uso para la gestión de usuarios

2. Gestión económica

    - Listar ofertas emitidas
    - Listar ofertas recibidas
    - Emitir oferta por futbolista
    - Aceptar oferta por futbolista
    - Rechazar oferta por futbolista
    - Cancelar oferta emitida por futbolista
    - Renovación de futbolistas
    - Fijar precio de venta de entradas

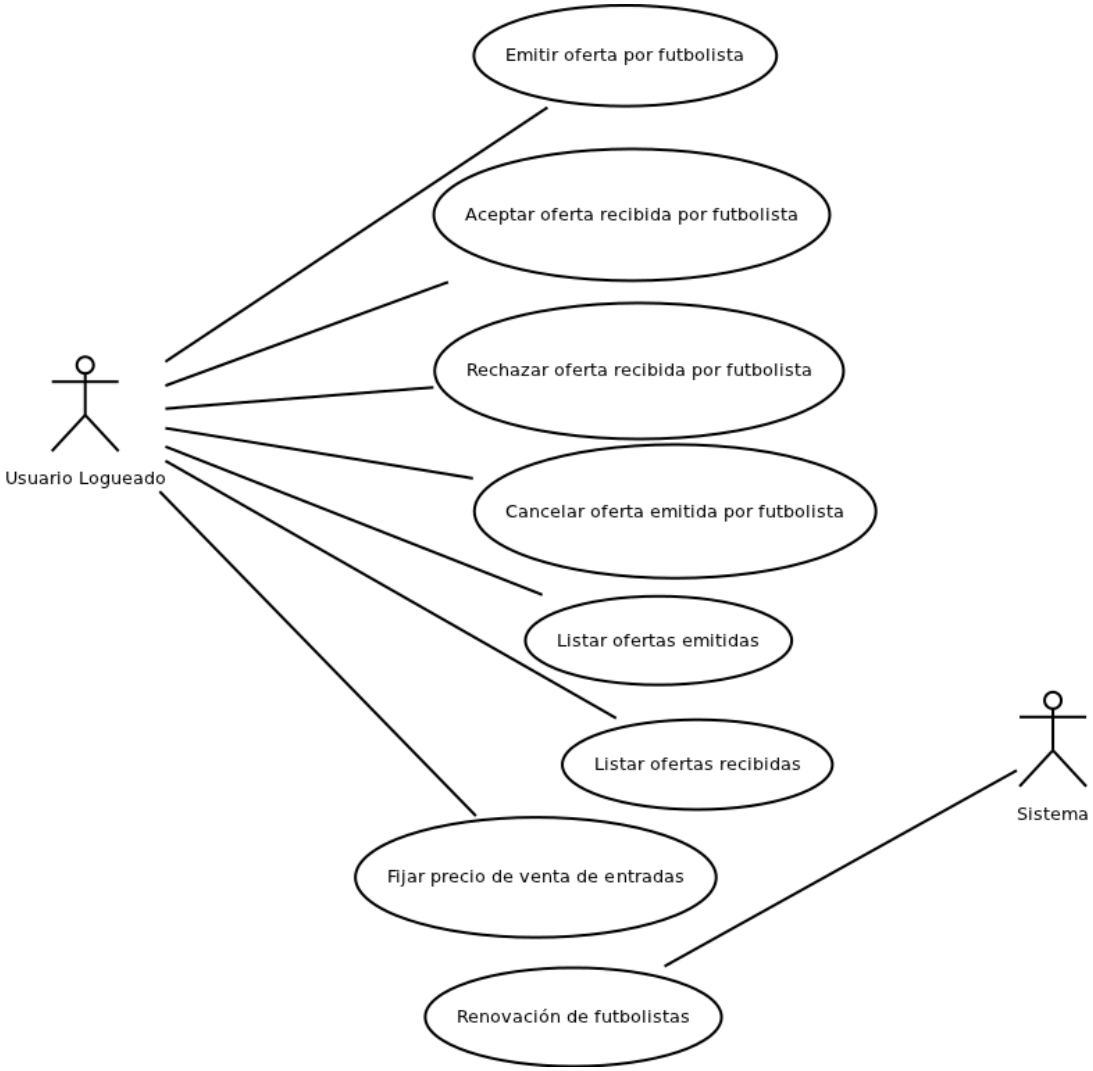

Figura 4.3: Diagrama de casos de uso para la gestión económica

3. Gestión deportiva
    - Planificación de alineaciones
    - Planificación de tácticas
    - Planificación de entrenamientos

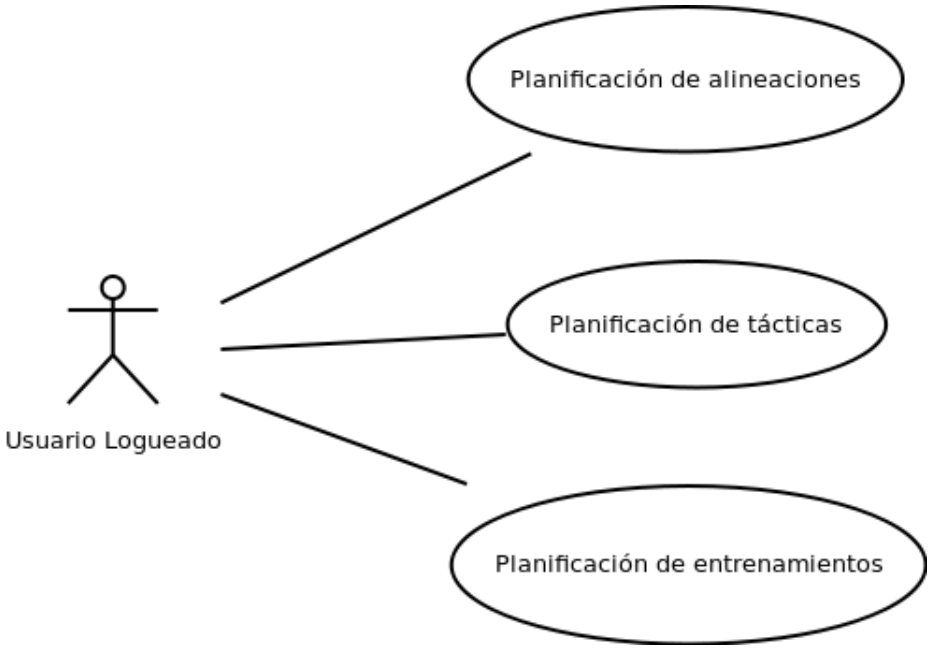

Figura 4.4: Diagrama de casos de uso para la gestión deportiva

4. Gestión de campeonatos
    - Asignar club
    - Simular jornadas
    - Comenzar jornadas
    - Pasar a siguiente jornada
    - Promoción y descenso de clubes
    - Composición de calendarios

Figura 4.5: Diagrama de casos de uso para la gestión de campeonatos

CAPÍTULO 5

Diseño

Para la implementación de toda la funcionalidad descrita en el capítulo anterior seguiremos las convenciones que el framework *Ruby on Rails* nos proporciona para una mayor velocidad y corrección en el desarrollo.

5.1. Arquitectura

Debido a *Ruby on Rails* es un framework que hace un uso intensivo del patrón *Modelo-Vista-Controlador* (MVC) [14].

El patrón *Modelo-Vista-Controlador* divide el software en tres capas bien diferenciadas que se adaptan muy bien a la tecnología Cliente-Servidor. Dichas capas son:

- **Modelo**: Esta capa es la relativa a la representación de la información con la que el proyecto interactuará. Está compuesto por las clases necesarias para el acceso, modificación, búsqueda y demás operaciones con los datos.
- **Vista**: En esta capa agruparemos todas las clases necesarias para la representación de la interfaz con la que nuestros usuarios interactuarán.
- **Controlador**: Será la capa que nos permita gestionar las peticiones que lleguen a nuestro servidor.

5.1.1. Capa Modelo

Para la capa de modelo y la consecuente persistencia de datos vamos a usar el *ORM* (mapeo objeto-relacional) que nos proporciona por defecto *Ruby On Rails*, la clase *ActiveRecord* que implementa el patrón *Active Record* pudiéndonos abstraer del *SGBD* relacional que usemos y las especificaciones únicas del lenguaje *SQL* que implemente el mismo.

5.1.2. Capa Vista

Para esta capa vamos a utilizar la clase *ActionView*. Dicha clase es una forma poderosa, sencilla y elegante, de embeber código *Ruby* dentro de cualquier tipo de fichero de la capa de presentación de nuestra página web (*JavaScript*, *CSS*).

Además, haremos uso de los frameworks *CoffeScript* [7] y *SASS* [5] para el desarrollo de los ficheros *JavaScript* y *CSS* respectivamente.

5.1.3. Capa Controlador

En esta capa haremos uso de la clase *ApplicationController*. Esta clase nos ayudará a modelar de forma sencilla y fácil el enrutador de nuestra aplicación y las respuestas que se hagan a través del cliente en la petición web.

Además, nos ayudará a establecer un flujo sencillo entre nuestras capas Vista y Modelo consiguiendo un menor acoplamiento y una mayor cohesión de nuestro sistema.

5.1.4. Active Record

Se basa en el patrón de mismo nombre el cual es un patrón de arquitectura que podemos encontrar en software que almacene información en bases de datos relacionales. Conforme a este patrón, un objeto que lo use habrá de integrar funciones para insertar, actualizar, eliminar y otras que correspondan en mayor o menor medida a las operaciones con columnas de la tabla de una base de datos.

Este patrón crea una interfaz que envuelve una tabla de una base de datos o una vista en una clase. En base a ello, un objeto de la clase corresponde a una fila en la tabla. La clase que envuelve a la tabla o vista implementa métodos o propiedades para acceder a cada columna.

Este patrón es normalmente usado para dar persistencia a los datos en un proyecto software, haciendo de *ORM* para la aplicación. Normalmente, las relaciones de clave foránea entre clases son implementadas como una propiedad del objeto [18].

Las funcionalidades que implementa la clase *Active Record* en *Rails* son:

- **Creación**: Permite la creación de nuevas líneas en la base de datos.

- **Condiciones**: Permite consultar datos en la base de datos usando para ello cadenas de texto o *hashes* que representen la parte *WHERE* en una sentencia SQL.
- **Sobrescribir métodos de accesibilidad por defecto**: Se pueden sobrescribir los métodos de accesibilidad para modificar el comportamiento en las operaciones de escritura, creación, lectura, y demás operaciones de lectura/escritura.
- **Métodos de consulta como atributos**: Se pueden crear nuevos métodos que se comporten como atributos para conocer si un atributo dentro de la tabla está inicializado.
- **Posibilidad de acceder a los atributos antes de que hayan pasado el casting de tipo**: Es una funcionalidad orientada sobre todo en las situaciones de validación.
- **Buscadores dinámicos en base al atributo**: Métodos estándar creados en base al nombre de los atributos correspondientes al nombre de la columna dentro de la tabla.
- **Serializar vectores, hashes y otros objetos en columnas de texto**: Es posible guardar este tipo de objetos en formato texto usando el lenguaje *YAML*.
- **Herencia simple de tabla**: Es posible el uso de la herencia simple entre tablas.
- **Conexión a múltiples bases de datos en diferentes modelos**: Es posible establecer diferentes conexiones a diferentes *SGBD* para diferentes clases.

5.1.5. Action View

Action View hace uso del sistema de plantillas *eRuby* para embeber código *Ruby* dentro de un documento de texto. Es ampliamente utilizado para dentro de documentos *HTML* y *JavaScript*.

5.1.6. Action Controller

Es el núcleo de una petición web en *Ruby on Rails*. Está formado por una o más acciones que son ejecutadas cuando se produzca una petición, justo después se puede redireccionar a otra acción o renderizar una plantilla. Una acción se define como un método público dentro del controlador, el cuál es automáticamente accesible al servidor web a través del enrutador de *Rails*.

Por defecto, sólo la clase *ApplicationController* dentro de una aplicación *Rails* hereda de la clase *ActionController:Base*. Los demás controladores heredarán de la clase *ApplicationController*. Este tipo de estructura nos proporciona una clase para configurar varias cosas que sean comunes a cualquier petición web.

Esta clase nos proporciona además las siguientes funcionalidades:

- **Peticiones**: El objeto de petición es totalmente accesible y se usa principalmente para conocer la cabecera de la petición *HTTP*.
- **Parámetros**: Todos los parámetros que lleva la petición, ya sean desde *GET* o *POST* o de la *URL*, son accesibles a través del *método* **params** que devolverá un *hash* uni o multidimensional con todos los parámetros asociados a la petición.

- **Sesiones**: Las sesiones permiten almacenar objetos entre petición y petición. Son bastante útiles para objetos que no estén aún preparados para pasar a ser almacenados en la base de datos. Para acceder a las sesiones bastará con llamar al método **session** y este devolverá un *hash* uni o multidimensional con las sesiones almacenadas.
- **Respuestas**: Toda acción lleva asociada una respuesta, la cual alberga las cabeceras y el documento para ser mandado al navegador del usuario. El objeto respuesta es generado automáticamente a través del uso de *renders* y redirecciones, con lo que no necesita la interacción del usuario.
- **Renderizaciones**: La clase *ActionController* manda contenido al usuario a través del uso de métodos de renderización. El más común es el uso de una plantilla para la renderización del contenido, a través de plantillas *ERB*.
- **Redirecciones**: Son usadas para moverse de una acción a otra.
- **Redirecciones y renderizaciones múltiples**: Es posible renderizar y redireccionar varias veces en la misma petición.

5.1.7. Diagrama de componentes y estructura

Podemos ver el diagrama de componentes en la figura 8.1.

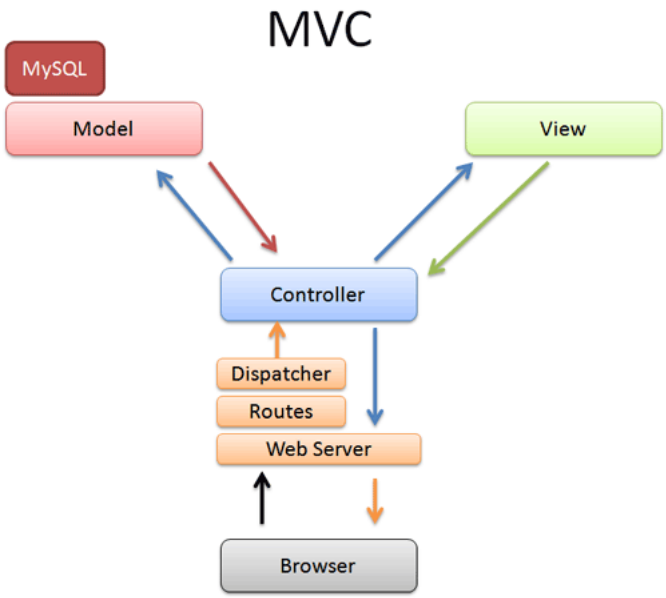

Figura 5.1: Diagrama de componentes

El framework *Ruby on Rails* hace especial hincapié en tres conceptos muy básicos que caracterizan toda su estructura, que son:

- **Convention over Configuration (COC)**: Convención frente a configuración.

- **Don't Repeat Yourself (DRY)**: No te repitas
- **Keep It Simple (KIS)**: Mantenlo simple.

Gracias a ello seguiremos la estructura de directorios en árbol para nuestro código fuente que genera *Rails* para una aplicación vacía.

- `/pfc_sfo/` raíz del proyecto *Rails*.
- `/pfc_sfo/app/` ficheros fuente principales.
- `/pfc_sfo/app/controllers/` ficheros fuente para los controladores.
- `/pfc_sfo/app/helpers/` ficheros fuente para los métodos de ayuda para refactorización de códigos en las vistas.
- `/pfc_sfo/app/mailers/` ficheros fuente para el código fuente de los correos que se envían desde el servidor.
- `/pfc_sfo/app/models/` ficheros fuente correspondientes a los modelos de datos.
- `/pfc_sfo/app/views/` ficheros fuente para las vistas de la aplicación, incluyendo como parte de las mismas las de los correos.
- `/pfc_sfo/app/assets/` ficheros fuente que han de ejecutarse en el navegador, archivos *CSS*, *JavasScript*, imágenes, etc...
- `/pfc_sfo/config/`, ficheros fuente de configuración de la aplicación web.
- `/pfc_sfo/config/environments/` ficheros fuente de configuración de los distintos entornos, desarrollo, test y producción.
- `/pfc_sfo/config/initializers/` ficheros de inicialización del servidor.
- `/pfc_sfo/config/locales/` ficheros *YAML* para los textos de traducción en diferentes idiomas.
- `/pfc_sfo/db/` ficheros autogenerados dónde se albergan la configuración de la Base de Datos.
- `/pfc_sfo/db/migrate/` ficheros para la manipulación de las tablas de la base de datos.
- `/pfc_sfo/lib/` ficheros para la extensión de la aplicación con otros subsistemas software.
- `/pfc_sfo/log/` ficheros log del servidor.
- `/pfc_sfo/public/` ficheros de acceso público a través del enrutador.
- `/pfc_sfo/script/` scripts para la generación automática de código y la automatización de procesos.
- `/pfc_sfo/test/` ficheros para almacenar test automáticos.

5.1.8. Entorno de ejecución

En la figura 8.2 podemos ver el diagrama de despliegue de la aplicación puesta en marcha en cualquiera de los entornos, ya sea este de desarrollo, test o producción.

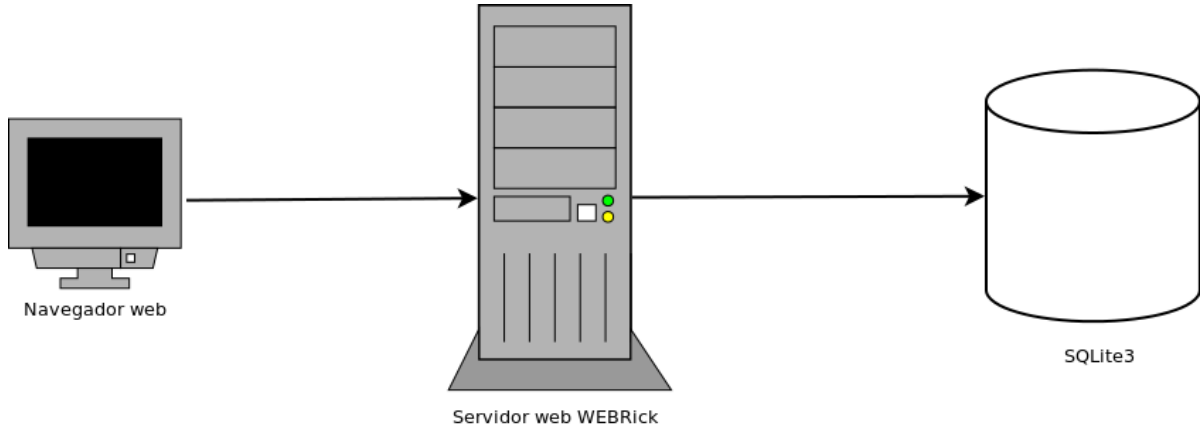

Figura 5.2: Diagrama de estructura

El sistema (servidor) interactuará con uno o más clientes web (navegadores). El servidor, a su vez estará formado por un servidor de aplicaciones web, en nuestro caso *Webrick*, y el sistema de gestión de base de datos será *SQLite3*.

Gracias a esta sencilla configuración no tendremos que configurar ninguna cuenta ni servicio en nuestro servidor web.

5.2. Diagramas de clases conceptuales

En la figura 8.3 podemos observar el diagrama de clases de la capa modelo de nuestra aplicación según la recolección de requisitos que hemos obtenido en el Capítulo 4 de esta memoria. Dichos diagramas serán realizados en notación *UML* [10].

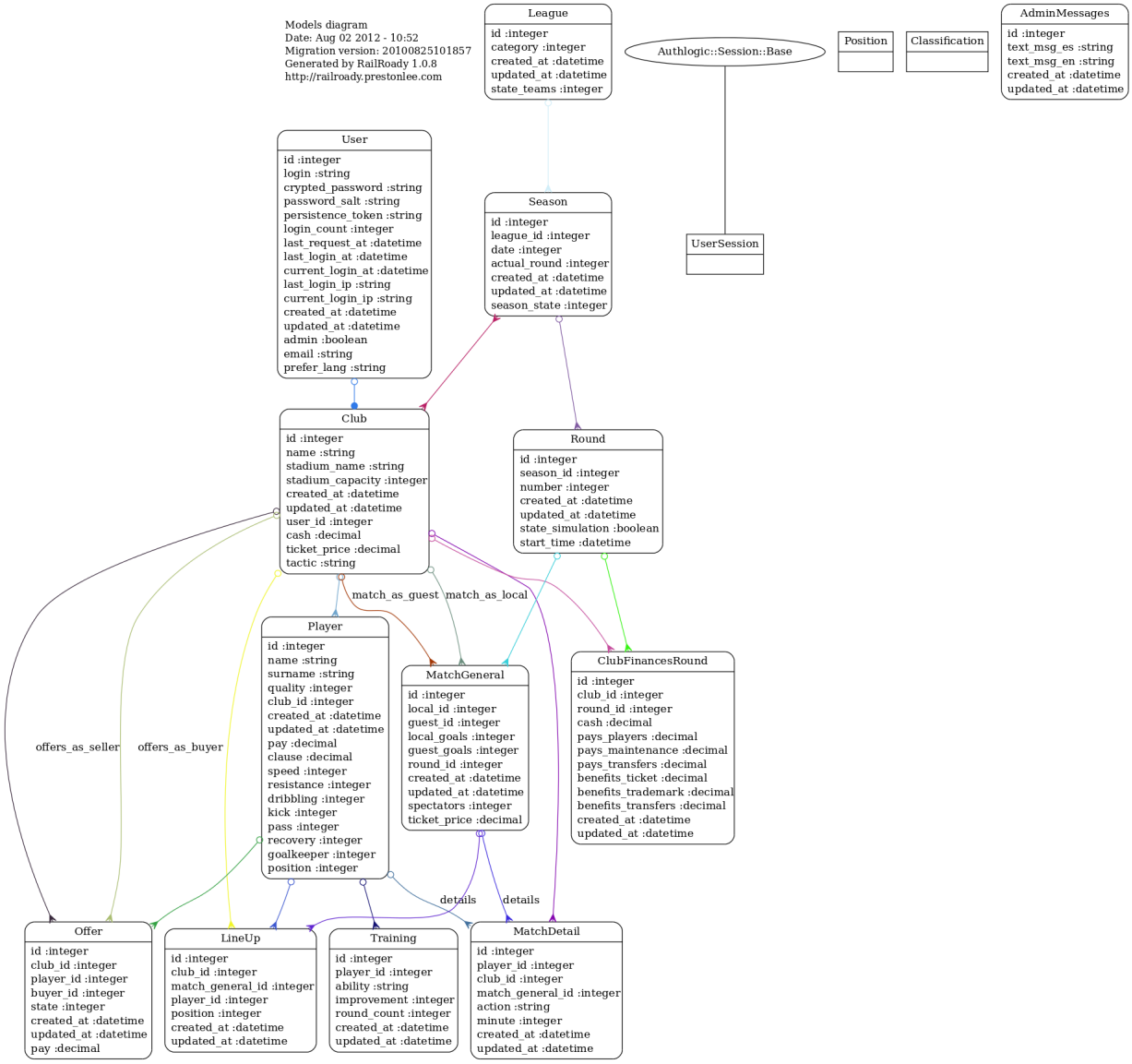

Figura 5.3: Diagrama de clases UML de la capa modelo

En la figura 8.4 podemos observar el diagramas de clases de la capa controlador de nuestra aplicación según la recolección de requisitos que hemos obtenido en el Capítulo 4.

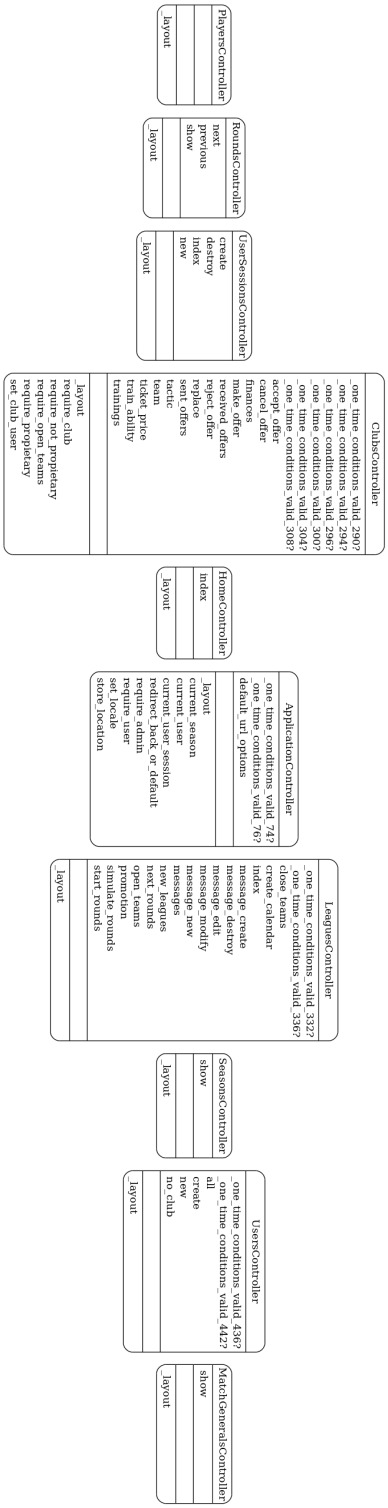

Figura 5.4: Diagrama de Clases UML de la capa controlador

5.3. Diagramas de secuencia de sistemas

En las siguientes secciones podremos observar los diagramas de secuencias del sistema software que hemos desarrollado.

Diagrama de secuencia de crear cuenta de usuario

En la figura 5.5 podemos observar como se construye un nuevo usuario dentro del sistema con los datos proporcionados por el usuario web.

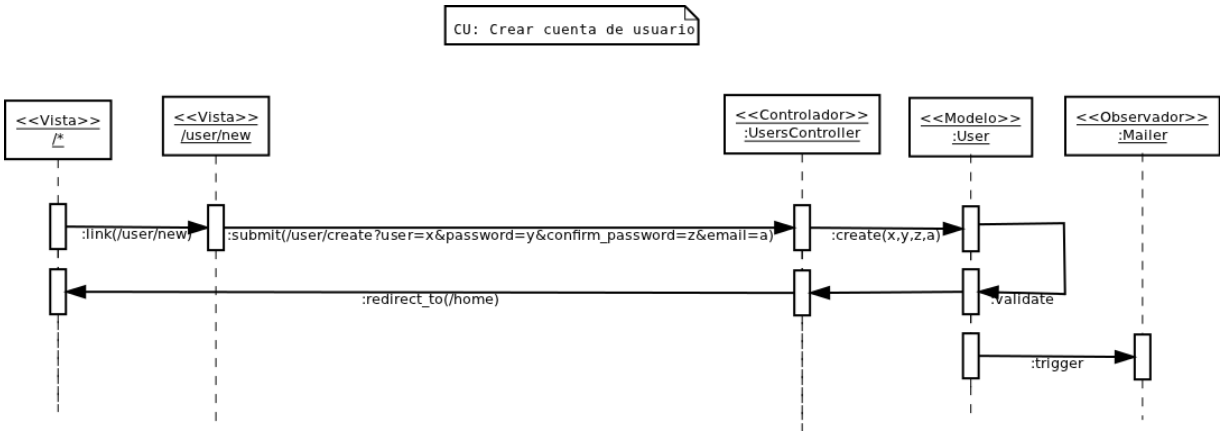

Figura 5.5: Diagrama de secuencia de crear cuenta de usuario

Diagrama de secuencia de acceder a cuenta de usuario

En la figura 5.6 podemos observar como se consigue acreditar las credenciales (usuario y contraseña) para iniciar sesión.

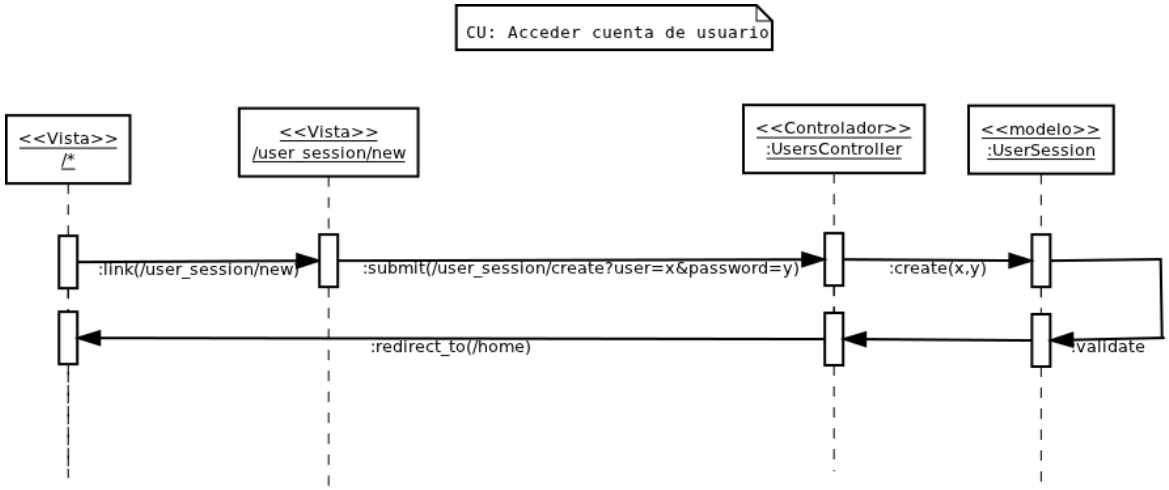

Figura 5.6: Diagrama de secuencia de acceder a cuenta de usuario

Diagrama de secuencia de abandonar cuenta de usuario

En la figura 5.7 podemos observar cómo se destruye la sesión iniciada por un usuario.

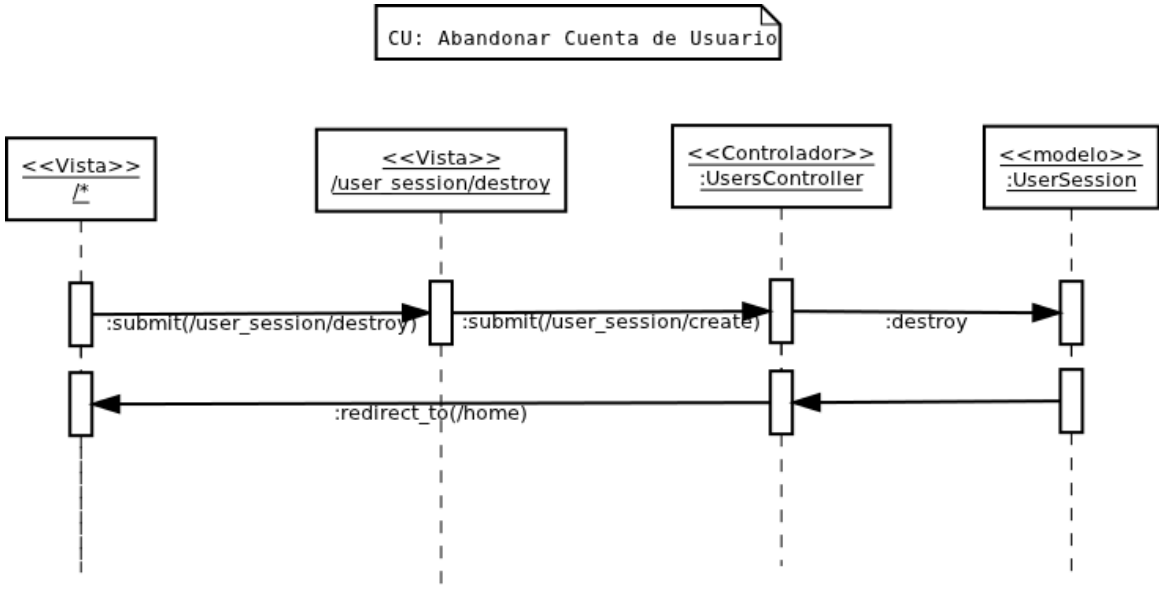

Figura 5.7: Diagrama de secuencia de abandonar cuenta de usuario

Diagrama de secuencia de abrir equipos

En la figura 5.8 cómo se establece el estado de las temporadas para permitir operaciones por parte de los usuarios en los equipos que poseen.

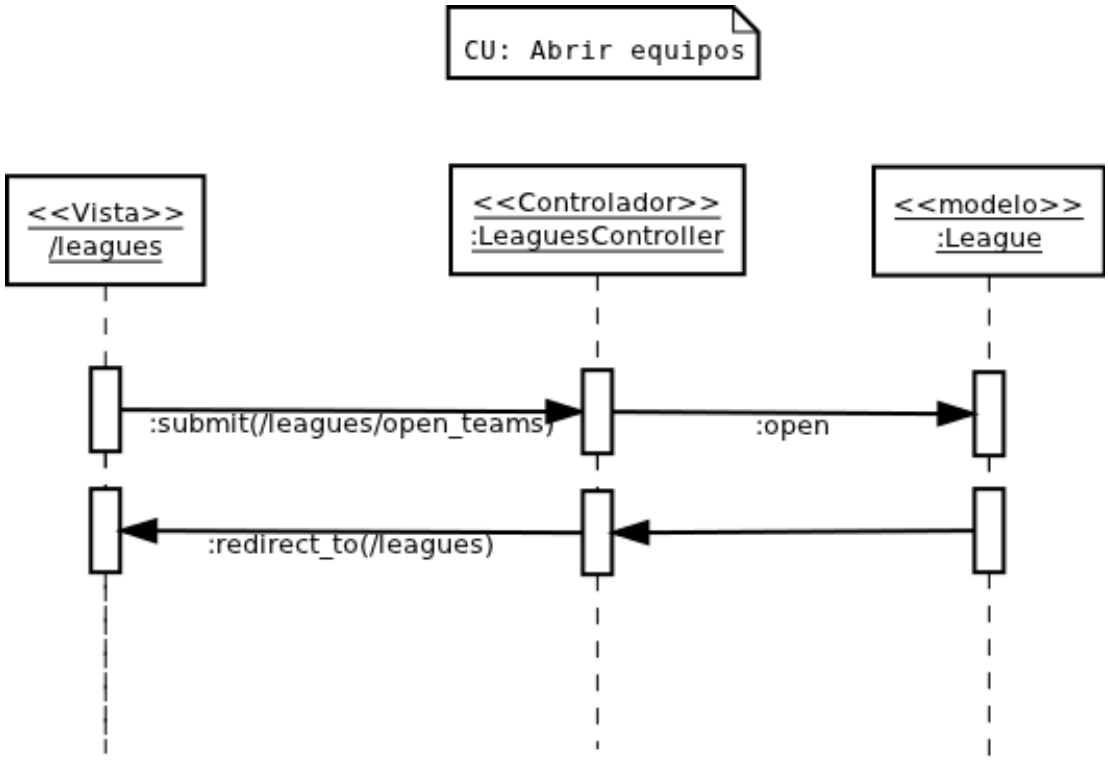

Figura 5.8: Diagrama de secuencia de abrir equipos

Diagrama de secuencia de cerrar equipos

En la figura 5.8 podemos observar cómo se establece el estado de las temporadas para impedir que los usuarios puedan realizar operaciones con sus equipos.

Este diagrama es igual al 5.8 pero con la ruta */leagues/close_teams* y la llamada al modelo con el método *close*.

Diagrama de secuencia de notificación de registro por e-mail

En la figura 5.9 se observa cómo se envía un correo al usuario una vez se ha registrado en la aplicación web.

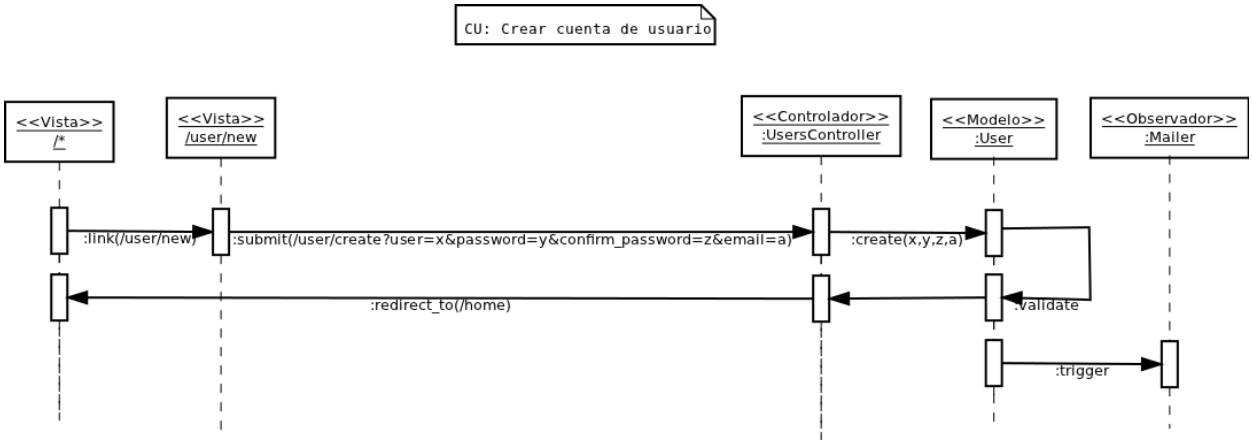

Figura 5.9: Diagrama de secuencia de notificación de registro por e-mail

Diagrama de secuencia de listar ofertas emitidas

En la figura 5.10 observamos cómo se establece el filtro de búsqueda para mostrar el listado de ofertas emitidas.

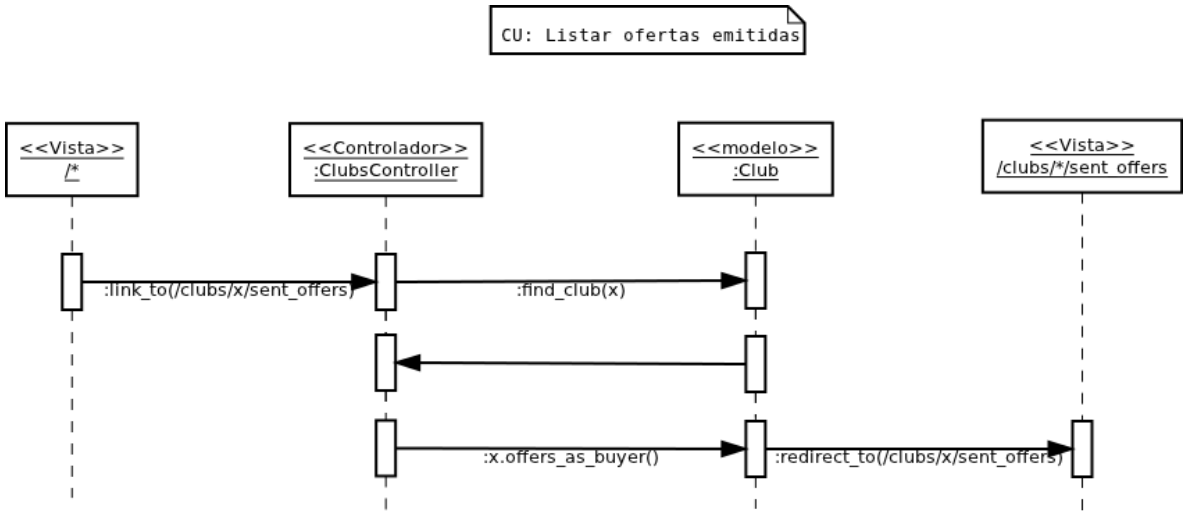

Figura 5.10: Diagrama de secuencia de listar ofertas emitidas

Diagrama de secuencia de listar ofertas recibidas

En la figura 5.10 observamos cómo se establece el filtro de búsqueda para mostrar el listado de ofertas recibidas.

Este diagrama es igual al 5.10 pero con la ruta */clubs/*/received_offers* y la llamada al modelo club con el método *offers_as_seller*.

Diagrama de secuencia de emitir oferta por futbolista

En la figura 5.11 se observa cómo se crea una nueva instancia de oferta y se relaciona con los usuarios y el futbolista involucrado en la misma.

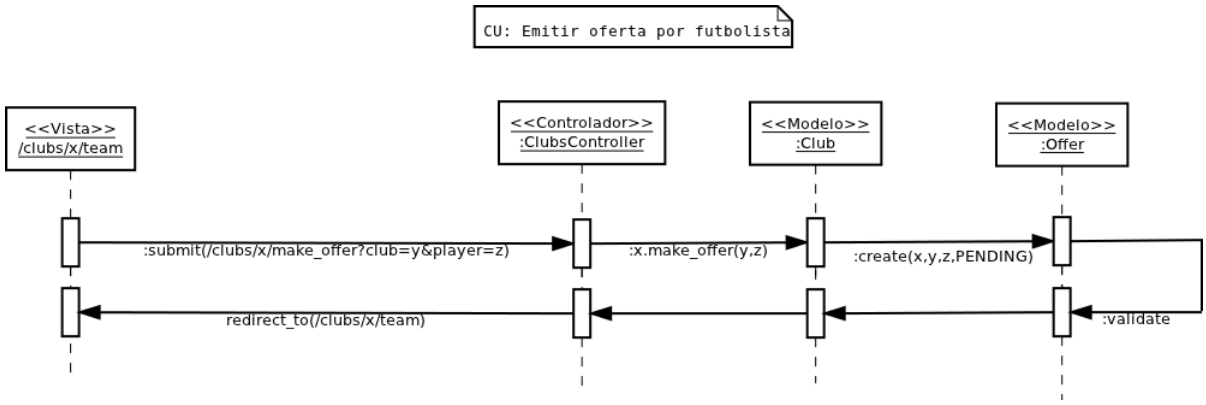

Figura 5.11: Diagrama de secuencia de emitir oferta por futbolista

Diagrama de secuencia de cancelar oferta emitida por futbolista

En la figura 5.12 se observa cómo se recupera la instancia de la oferta emitida y se modifica su estado para cancelarla.

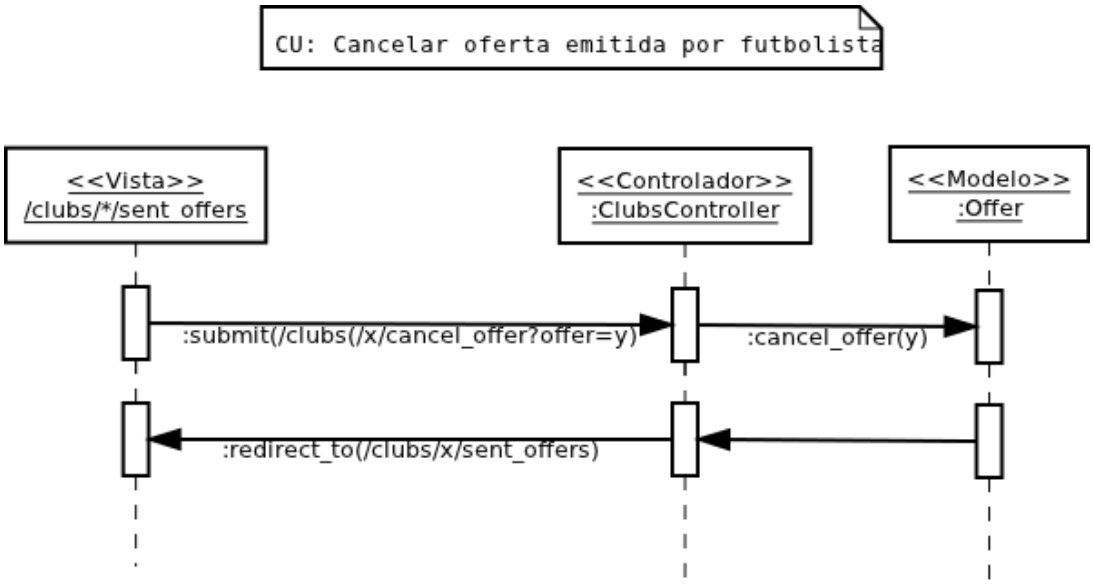

Figura 5.12: Diagrama de secuencia de cancelar oferta emitida por futbolista

Diagrama de secuencia de rechazar oferta recibida por futbolista

En la figura 5.13 se observa cómo se recupera la instancia de la oferta recibida y se modifica su estado para rechazar la misma.

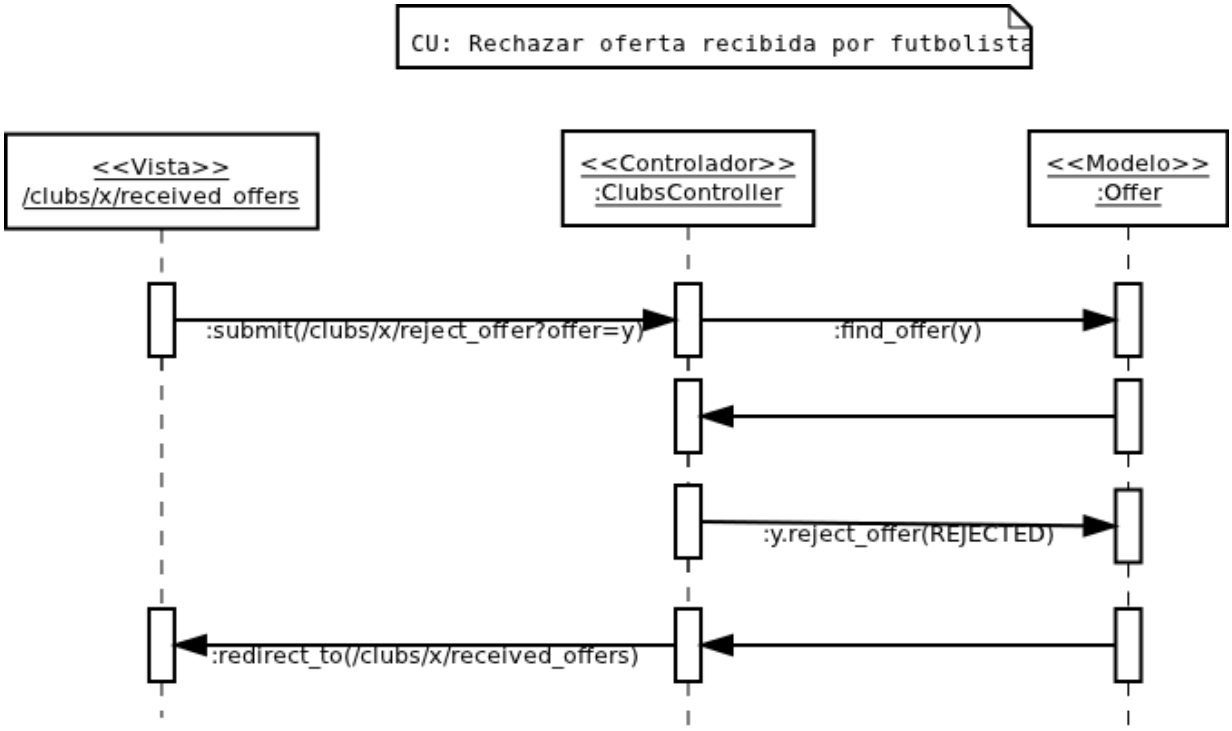

Figura 5.13: Diagrama de secuencia de rechazar oferta recibida por futbolista

Diagrama de secuencia de aceptar oferta recibida por futbolista

En la figura 5.14 se observa cómo se recupera la instancia de la oferta recibida y se actualizan los modelos de datos involucrados (futbolista, club y usuarios) para llevar a cabo la transferencia del futbolista entre los clubes.

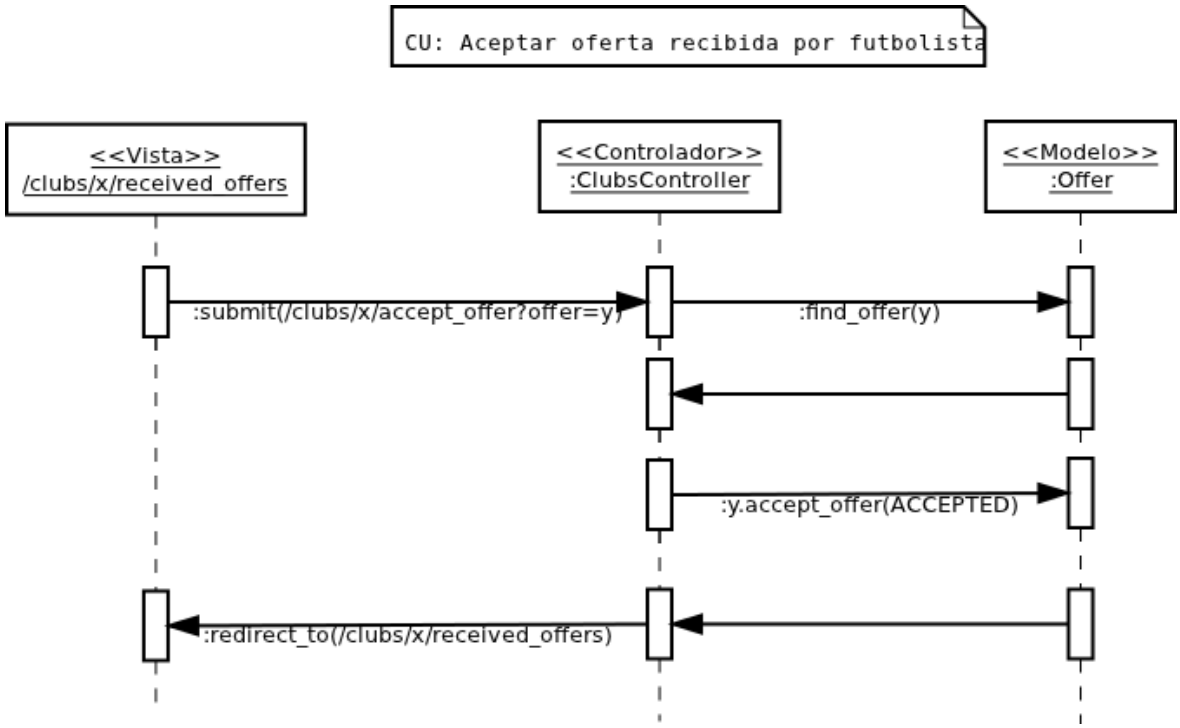

Figura 5.14: Diagrama de secuencia de aceptar oferta recibida por futbolista

Diagrama de secuencia de renovación de futbolistas

En la figura 5.15 se describe cómo se renuevan automáticamente los futbolistas para las siguientes temporadas actualizando sus atributos.

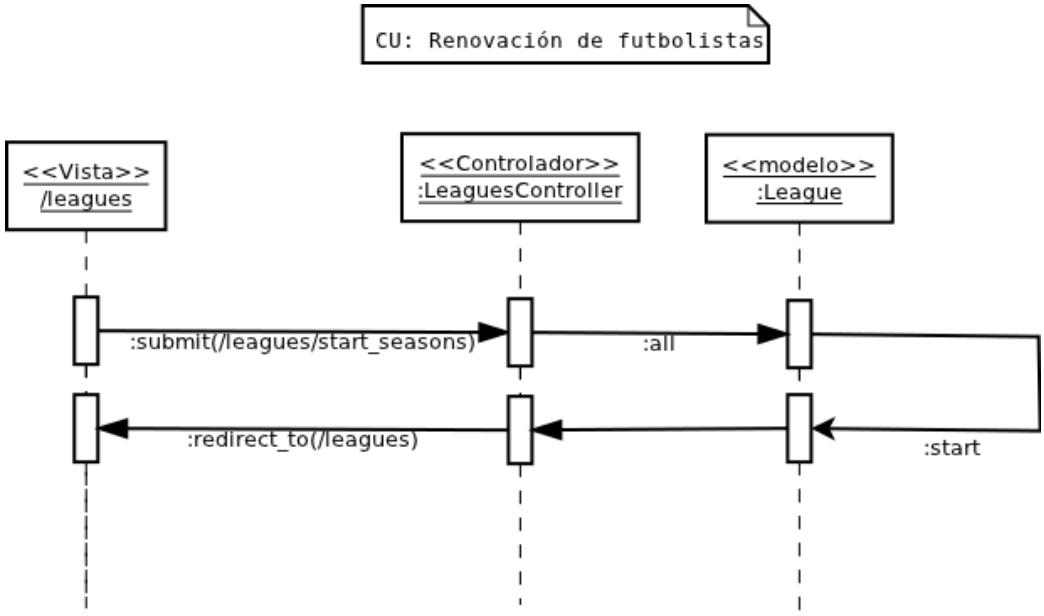

Figura 5.15: Diagrama de secuencia de renovación de futbolistas

El proceso de renovación está integrado en start.

Diagrama de secuencia de fijar precio de venta de entradas

En la figura 5.16 se describe cómo se modifica el atributo del precio de venta de entradas.

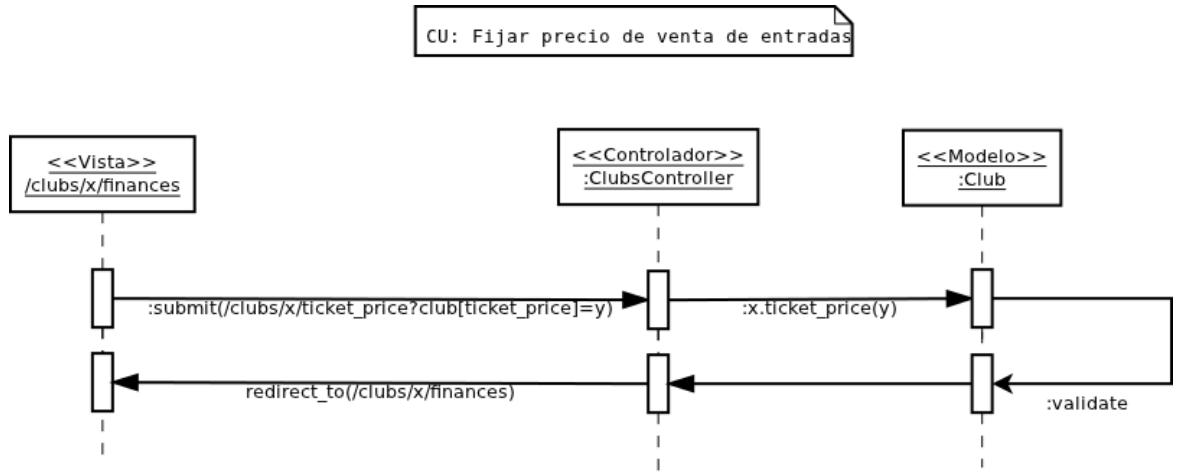

Figura 5.16: Diagrama de secuencia de fijar precio de venta de entradas

Diagrama de secuencia de planificación de alineaciones

En la figura 5.17 se describe cómo se cambian las alineaciones para el próximo partido a través de la modificación del atributo de posición.

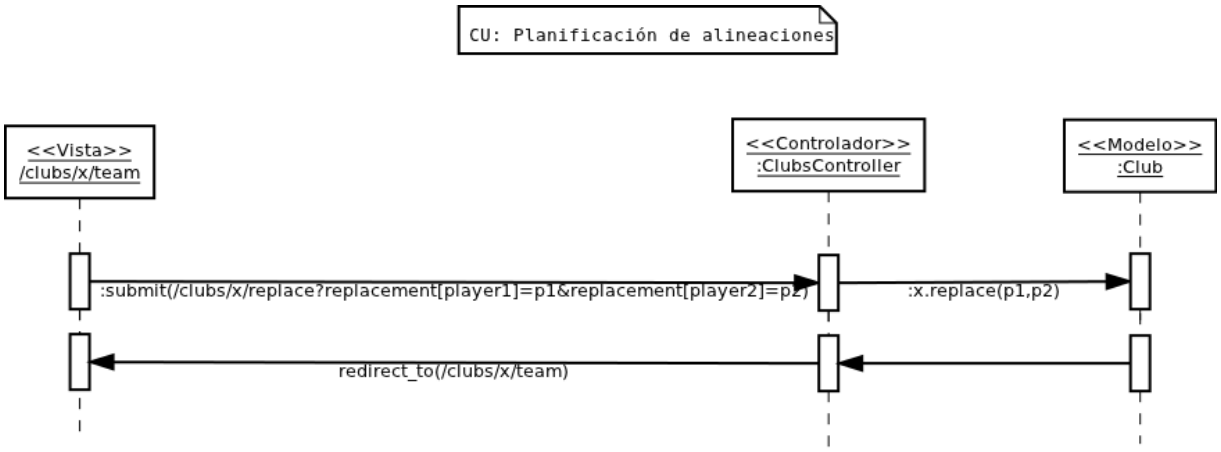

Figura 5.17: Diagrama de secuencia de emitir planificación de alineaciones

Diagrama de secuencia de planificación de tácticas

En la figura 5.18 se muestra cómo se cambia la táctica usada para el próximo partido a través de la modificación del atributo de táctica del club.

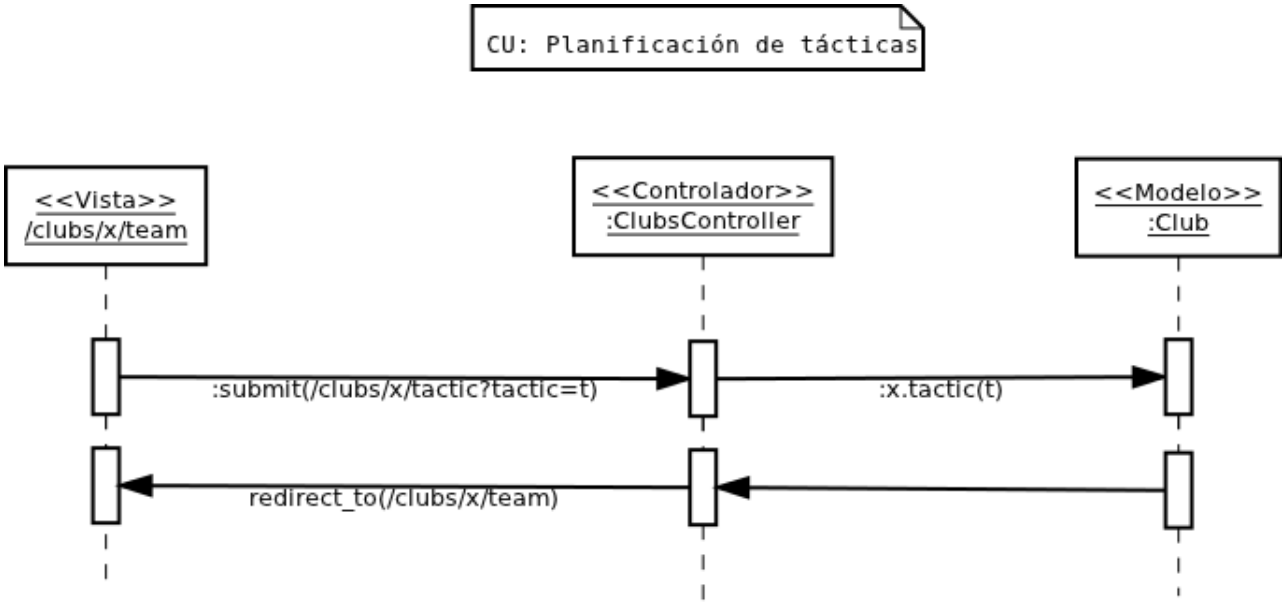

Figura 5.18: Diagrama de secuencia de planificación de tácticas

Diagrama de secuencia de planificación de entrenamientos

En la figura 5.19 se muestra cómo se crean nuevas instancias para la planificación de los entrenamientos de los futbolistas asociados a un club.

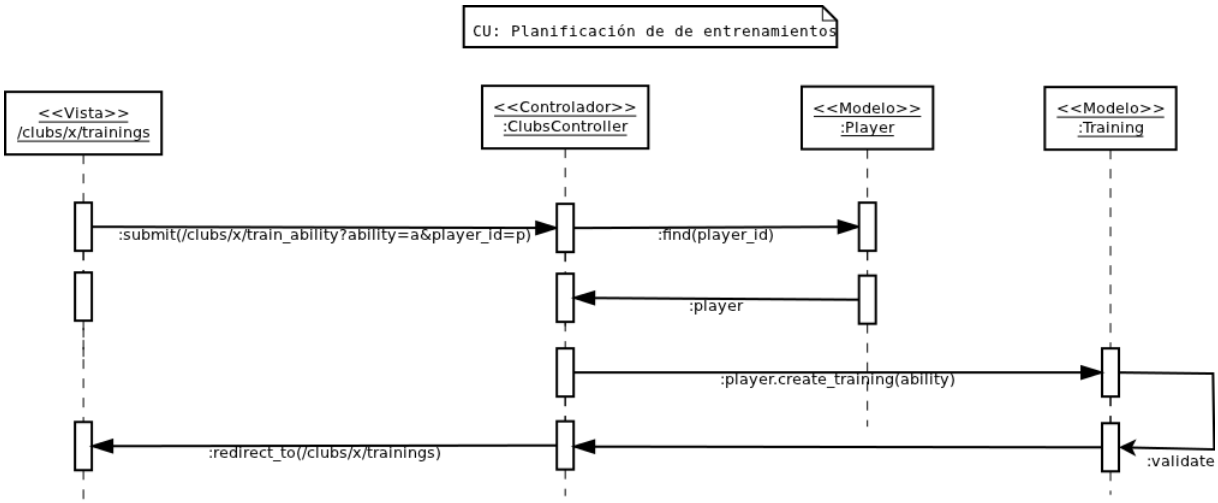

Figura 5.19: Diagrama de secuencia de planificación de entrenamientos

Diagrama de secuencia de asignar club

En la figura 5.20 se observa cómo se crea una nueva instancia de un club para todo usuario que no tenga club asignado.

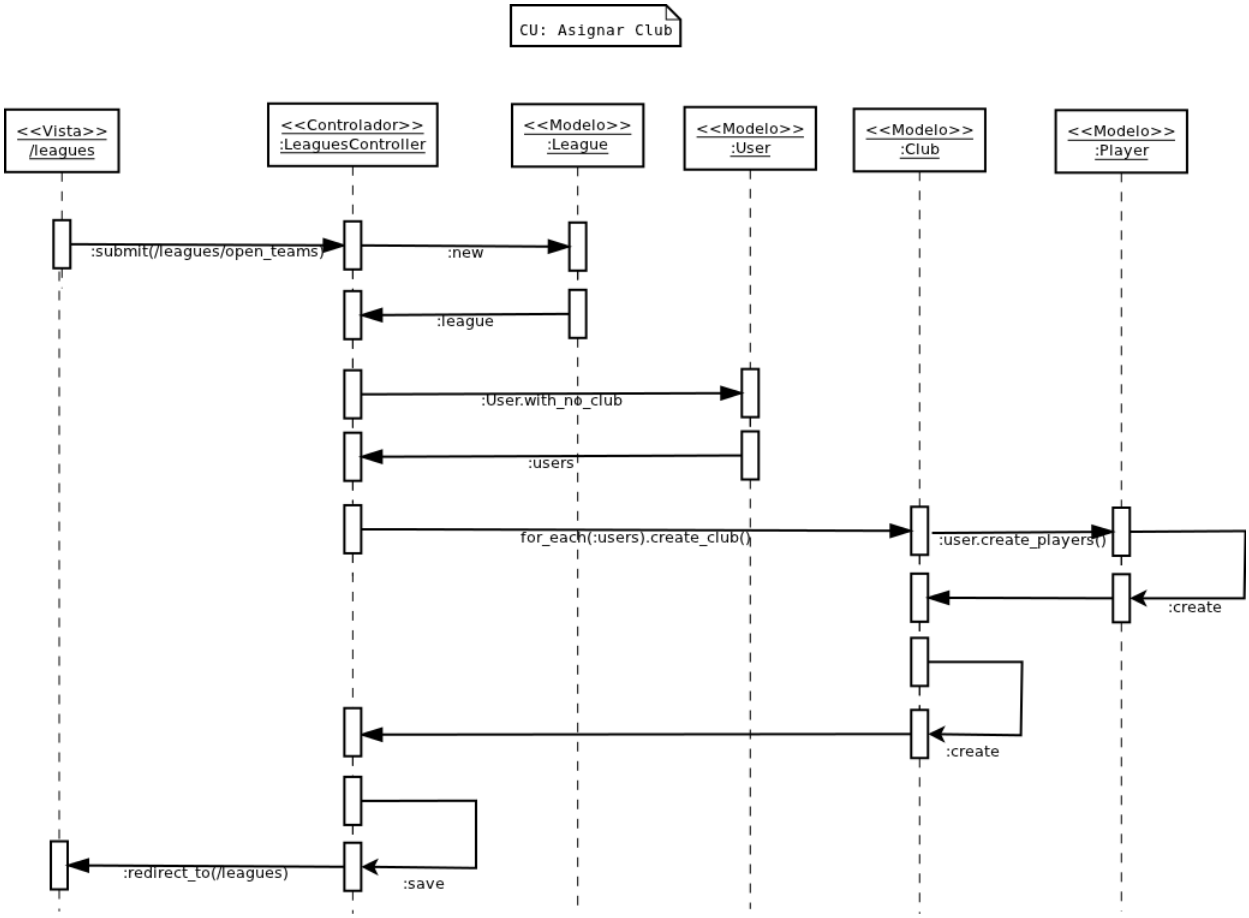

Figura 5.20: Diagrama de secuencia de secuencia de asignar club

Diagrama de secuencia de simular jornadas

En la figura 5.21 se observa cómo se crean nuevas instancias de detalles de partidos por cada uno de los partidos no jugados y pertenecientes a la jornada actual del sistema.

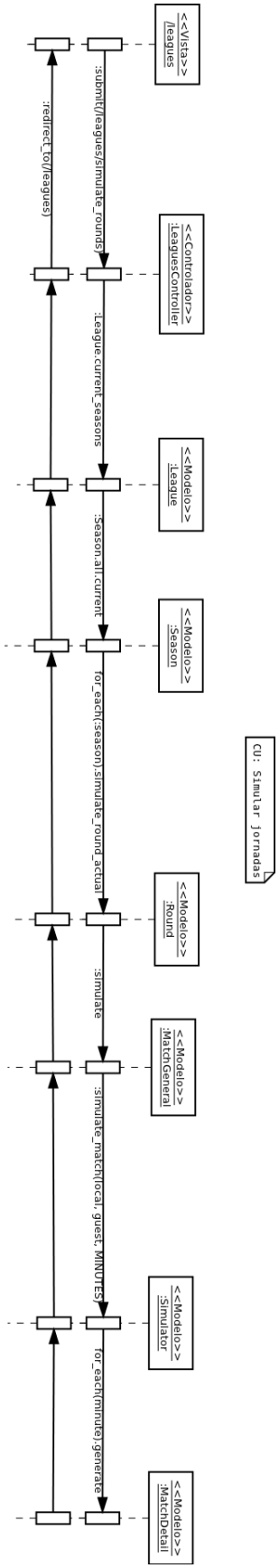

Figura 5.21: Diagrama de secuencia de simular jornadas

Diagrama de secuencia de comenzar jornadas

En la figura 5.22 se observa cómo se actualiza el atributo correspondiente de todos los partidos actuales para estipular que el partido ha comenzado a la fecha actual del sistema.

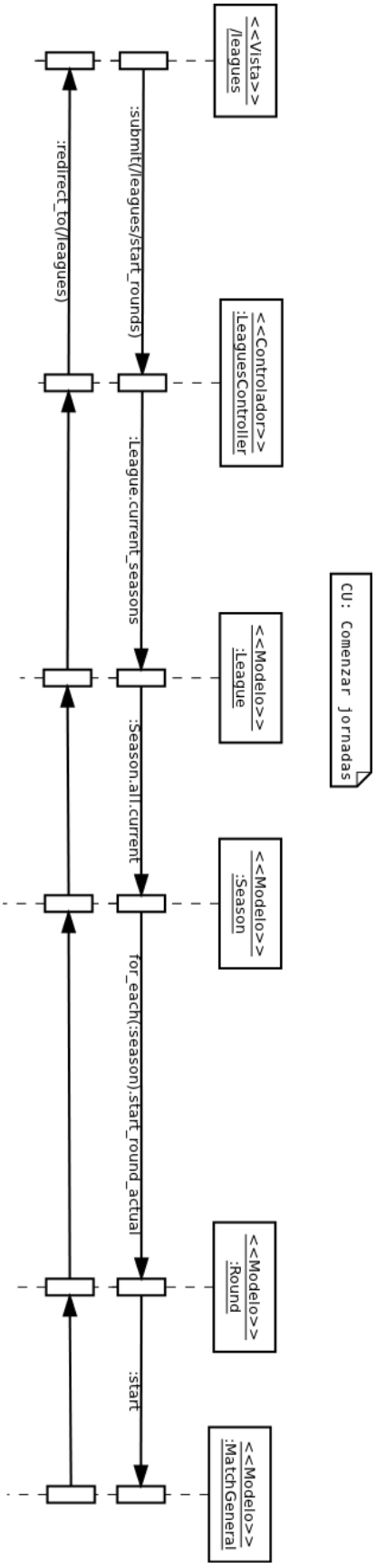

Figura 5.22: Diagrama de secuencia de comenzar jornadas

Diagrama de secuencia de pasar a siguiente jornada

En la figura 5.23 se detalla cómo se actualizan los modelos de datos de temporada, jornada, jugador, entrenamiento, finanzas y club para avanzar en la competición.

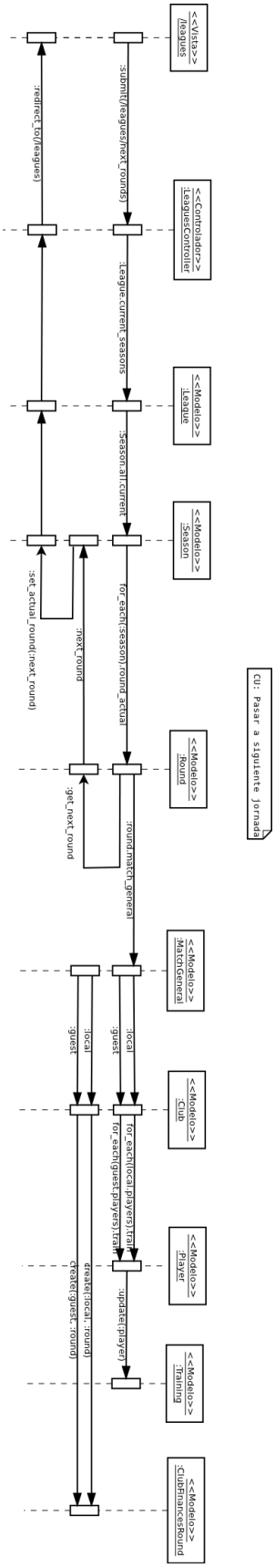

Figura 5.23: Diagrama de secuencia de pasar a siguiente jornada

Diagrama de secuencia de promoción y descenso de clubes

En la figura 5.24 se observa cómo se crean nuevas instancias de nuevas temporadas y se modifican los estados de las temporadas actuales para simular el avance en la competición y el progreso y descenso jerárquico en las divisiones del juego.

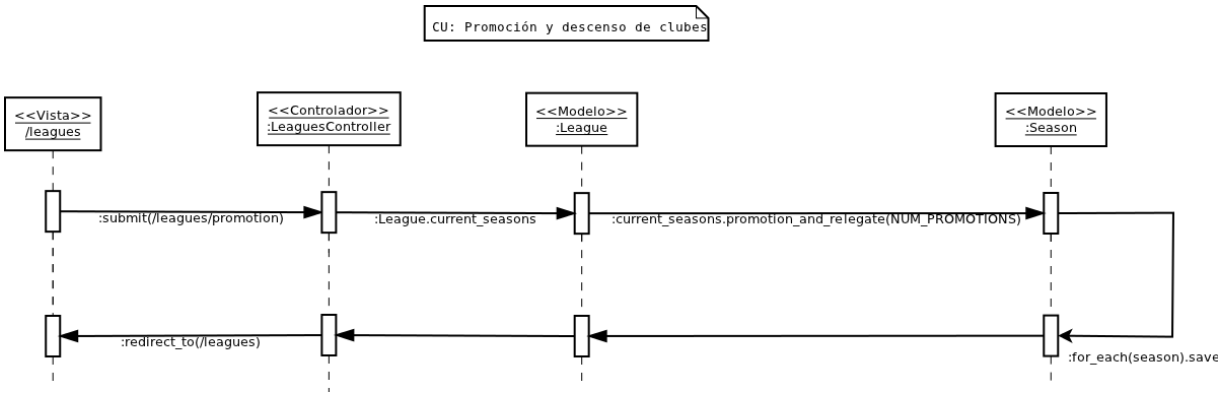

Figura 5.24: Diagrama de secuencia de promocion y descenso de clubes

Diagrama de secuencia de composición de calendarios

En la figura 5.25 se observa cómo se crean nuevas instancias de las jornadas que estarán relacionadas con los clubes en los que se coincida en las mismas divisiones.

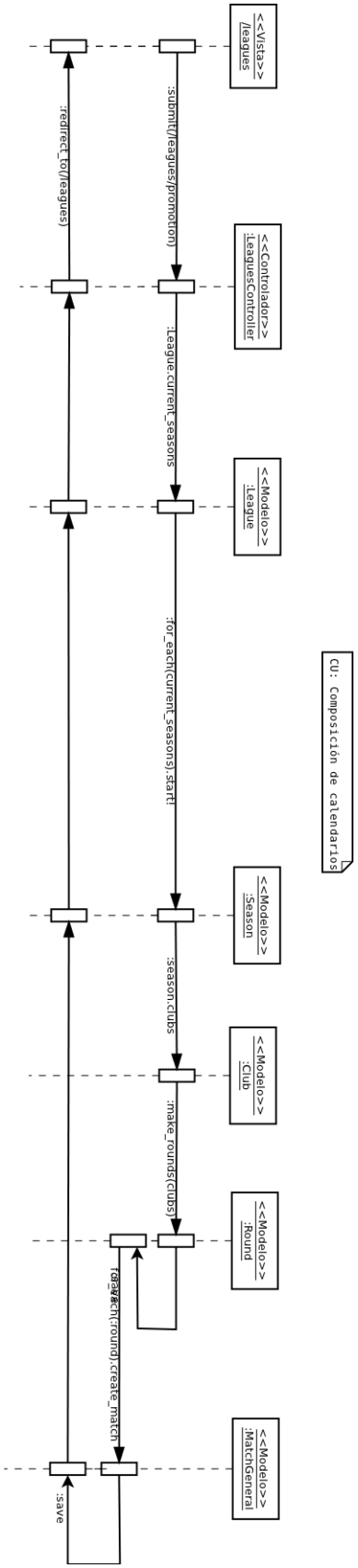

Figura 5.25: Diagrama de secuencia de composición de calendarios

5.4. Sistema de simulación

Este apartado de la memoria requiere una especial mención; ya que es el sistema que hará que el usuario tenga la percepción de que los cambios y modificaciones a la táctica, alineación, entrenamientos y precio de entradas que realice en su equipo, tengan un reflejo directo en el desarrollo de los partidos que conforman la temporada que disputa.

Para perfilar el sistema de simulación de nuestro proyecto de fin de carrera tuvimos varias reuniones con nuestros tutores y acotamos la forma de simular un partido en base a la táctica y alineación que presentan los clubes que disputarán el partido.

A continuación mostraremos la forma en que nuestro algoritmo arroja un resultado para el partido.

5.4.1. Número de jugadas

5.4.1.1. Cálculo general

El sistema de simulación hará varios cálculos para conformar cuantas serán las jugadas de ataque de las que dispondrá cada uno de los clubes en base a la calidad ponderada de los atributos que los jugadores poseen en la táctica seleccionada por el usuario.

Dicha forma de realizar la ponderación se estipulará en un fichero yml como el siguiente, perteneciente en este caso a la táctica de fútbol 4-4-2:

```
name: 4-4-2
defense: 4
possesion: 4
attack: 2
image: tactics/4-4-2.png
players:
  pos1:
    sym: GK
    goalkeeper: 1
    speed: 0
    resistance: 0
    recovery: 0
    pass: 0
    dribbling: 0
    kick: 0
  pos2:
    sym: LL
    goalkeeper: 0
    speed: 0.30
    resistance: 0.30
    recovery: 0.20
    pass: 0.10
    dribbling: 0.10
    kick: 0
  pos3:
```

```
    sym: LC
    goalkeeper: 0
    speed: 0.25
    resistance: 0.25
    recovery: 0.40
    pass: 0.10
    dribbling: 0
    kick: 0
  pos4:
    sym: RC
    goalkeeper: 0
    speed: 0.25
    resistance: 0.25
    recovery: 0.40
    pass: 0.10
    dribbling: 0
    kick: 0
  pos5:
    sym: RL
    goalkeeper: 0
    speed: 0.30
    resistance: 0.30
    recovery: 0.20
    pass: 0.10
    dribbling: 0.10
    kick: 0
  pos6:
    sym: CM
    goalkeeper: 0
    speed: 0.20
    resistance: 0.20
    recovery: 0.30
    pass: 0.20
    dribbling: 0.10
    kick: 0
  pos7:
    sym: ML
    goalkeeper: 0
    speed: 0.35
    resistance: 0.15
    recovery: 0.05
    pass: 0.10
    dribbling: 0.25
    kick: 0.10
  pos8:
    sym: MR
    goalkeeper: 0
    speed: 0.35
    resistance: 0.15
    recovery: 0.05
    pass: 0.10
    dribbling: 0.25
    kick: 0
  pos9:
    sym: MP
    goalkeeper: 0
```

```
    speed: 0.20
    resistance: 0.20
    recovery: 0
    pass: 0.30
    dribbling: 0.10
    kick: 0.20
  pos10:
    sym: FL
    goalkeeper: 0
    speed: 0.25
    resistance: 0.15
    recovery: 0
    pass: 0.15
    dribbling: 0.20
    kick: 0.25
  pos11:
    sym: FR
    goalkeeper: 0
    speed: 0.25
    resistance: 0.15
    recovery: 0
    pass: 0.15
    dribbling: 0.20
    kick: 0.25
```

En dicho fichero podemos ver la ponderación de cada atributo que damos a un jugador según la posición que éste ocupe en el sistema táctico, la cual hemos denominamos en el sistema como *calidad táctica*.

Una vez tengamos calculada la *calidad táctica* de todos los integrantes del once inicial en nuestro equipo nos dispondremos a sacar la media de dichas calidades y asignaremos un número de jugadas a cada equipo con un máximo de 88 jugadas por partido.

Esto quiere decir que si se enfrentan dos equipos cuyos onces iniciales tienen la misma media, por ejemplo 50, cada equipo dispondrá de 44 jugadas de ataque, las cuales podrán ser contrarrestadas si con la táctica seleccionada se consigue robar jugadas de ataque o neutralizar las jugadas de gol.

5.4.1.2. Jugada de ataque

Una vez que hemos calculado el número de jugadas de ataque del que disponemos, hemos de simular qué tipo de jugada va tener lugar en el desarrollo del partido. Para ello se hará uso de la posibilidad de emular la tirada de un dado a través de la generación de un número aleatorio para reflejar la aleatoriedad y la suerte que existe en la vida real a la hora de jugar un partido de fútbol.

En nuestro sistema hemos decidido que existan 15 tipos de jugadas de ataque, entre ellas, la que marca la diferencia a la hora de simular un partido, el gol.

Por tanto, podemos imaginar nuestro sistema de simulación como un dado de 15 caras en el cual existe una de ellas correspondiente a marcar gol, de esta forma tenemos un quinceavo de posibilidades por jugada de marcar. Así conseguimos que el usuario sienta que es importante entrenar, posicionar e intentar

la compra de jugadores que se adapten de forma correcta en la táctica seleccionada.

5.4.1.3. Neutralizar jugadas de gol

Una vez calculadas el número de jugadas, será posible resistir una posible jugada de gol dependiendo de la suma de *calidades tácticas* de los jugadores situados en defensa para el equipo que sufre la jugada de gol, respecto a la suma de calidades *tácticas* de los jugadores atacantes del equipo que está haciendo uso de la jugada de gol.

De esta forma se consigue emular la dificultad que supondría el marcar un gol para un equipo que utiliza una táctica estándar 4-4-2 contra un equipo que utiliza una táctica 5-4-1, respetando aún así la calidad de dichos jugadores, ya que será necesario que los jugadores dispuestos en la táctica tengan una buena calidad para conseguir una buena suma y así disponer de más facilidad para neutralizar jugadas de gol.

5.4.1.4. Robar jugadas de ataque

De la misma forma que es posible neutralizar jugadas de gol, para contemplar el beneficio de tener un centro del campo poblado, el sistema sumará las calidades de los jugadores dispuestos en el medio campo de ambos equipo y los ponderará de tal forma que se le roben algunas jugadas de ataque al equipo que presente una suma menor en las *calidades tácticas* de los jugadores situados en el centro del campo.

5.4.2. Número de espectadores

El sistema simulará el número de espectadores que acudieron al estadio en base a la posición que mantengan los equipos en la tabla clasificatoria, así como la calidad que posean los equipos que disputan el partido, todo ello ponderado en base al precio de venta de las entradas para emular la afluencia al estadio de forma realista.

CAPÍTULO 6

Implementación

En este capítulo vamos a describir y explicar los detalles de implementación del sistema software; además de comentar los problemas y contingencias surgidas durante el proceso de desarrollo del mismo.

Para situar mejor al lector pasaremos a descargar e instalar el código fuente base de la aplicación desarrollada que será descargable desde [2].

El código fuente está albergado en un repositorio de tipo *GIT*, aunque también será descargable como un *tarball* a través de la interfaz web.

Para que la aplicación sea totalmente funcional habrá que añadir los siguientes directorios al árbol base de la aplicación:

- `/tmp/`
- `/log/`

Para que el sistema pueda ser ejecutado, además se habrán de cumplir los requisitos especificados en 3.3. También será importante tener configurado un SGBD relacional a escoger entre:

- MySQL
- Postgres
- SQLite3
- Oracle

Una vez que tengamos el código fuente aislado en un directorio habremos de ejecutar los siguientes comandos en orden:

1. `gem install`
2. `rake db:create`
3. `rake db:migrate`
4. `rails s`

Con todos estos pasos dados ya podremos acceder a nuestra aplicación web a través del navegador y de la dirección `http://localhost:3000/`.

6.1. Modelo de datos

Al usar como gestor *ORM*, la clase incorporada en *Rails*, *ActiveRecord*, habrá que especificar los denominados ficheros de migración para ir creando las tablas a través de código ruby. Vemos aquí un ejemplo de uno de los archivos de migración extraídos de nuestro proyecto de fin de carrera:

```ruby
class CreatePlayers < ActiveRecord::Migration
  def self.up
    create_table :players do |t|
      t.string :name, :limit => 20, :null => false
      t.string :surname, :limit => 30, :null => false
      t.integer :quality, :null => false
      #t.integer :age, :null => false
      t.integer :club_id, :null => true

      t.timestamps
    end
  end

  def self.down
    drop_table :players
  end
end
```

Como podemos observar, a través de los bloques de Ruby creamos comandos para inicializar y revertir los cambios que traen la migración consigo. Los cambios que podemos realizar entre otras son:

- Crear tablas.
- Eliminar tablas.
- Crear columnas.
- Eliminar columnas.

- Modificar nombres de tablas.
- Modificar nombres de columnas.
- Insertar nuevos datos.
- Eliminar datos.
- Migrar datos.

El conjunto de ficheros de migración crearán el archivo `schema.rb` que tiene una definición completa de las tablas que albergan nuestra base de datos, que podrá consultarse en el anexo D

6.2. Controlador de versiones

Para controlar los diversos cambios que realizamos sobre el producto o la configuración del mismo hemos decidido usar *GIT* el cuál se integra a la perfección con nuestro entorno de desarrollo *Linux* y además es uno de los controladores de la versión más en boga del actual panorama software.

El sistema de versiones *GIT* nos proporciona las siguientes funcionalidades [20]:

- Fuerte apoyo al desarrollo no-lineal, por ende rapidez en la gestión de ramas y mezclado de diferentes versiones. *Git* incluye herramientas específicas para navegar y visualizar un historial de desarrollo no-lineal. Una presunción fundamental en *Git* es que un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente, conforme se pasa entre varios programadores que lo revisan.
- Gestión distribuida. Al igual que *Darcs*, *BitKeeper*, *Mercurial*, *SVK*, *Bazaar* y *Monotone*, *Git* le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales. Los cambios se importan como ramas adicionales y pueden ser fusionados en la misma manera que se hace con la rama local.
- Los almacenes de información pueden publicarse por *HTTP*, *FTP*, *rsync* o mediante un protocolo nativo, ya sea a través de una conexión *TCP/IP* simple o a través de cifrado *SSH*. *Git* también puede emular servidores *CVS*, lo que habilita el uso de clientes *CVS* pre-existentes y modulos *IDE* para *CVS* pre-existentes en el acceso de repositorios *Git*.
- Los repositorios *Subversion* y *svk* se pueden usar directamente con *git-svn*.
- Gestión eficiente de proyectos grandes, dada la rapidez de gestión de diferencias entre archivos, entre otras mejoras de optimización de velocidad de ejecución.
- Todas las versiones previas a un cambio determinado, implican la notificación de un cambio posterior en cualquiera de ellas a ese cambio (denominado autenticación criptográfica de historial). Esto existía en *Monotone*.
- Resulta algo más caro trabajar con ficheros concretos frente a proyectos, eso diferencia el trabajo frente a *CVS*, que trabaja con base en cambios de fichero, pero mejora el trabajo con afectaciones de código que concurren en operaciones similares en varios archivos.

- Los renombrados se trabajan basándose en similitudes entre ficheros, aparte de nombres de ficheros, pero no se hacen marcas explícitas de cambios de nombre con base en supuestos nombres únicos de nodos de sistema de ficheros, lo que evita posibles, y posiblemente desastrosas, coincidencias de ficheros diferentes en un único nombre.

- Realmacenamiento periódico en paquetes (ficheros). Esto es relativamente eficiente para escritura de cambios y relativamente ineficiente para lectura si el reempaquetado (con base en diferencias) no ocurre cada cierto tiempo.

A continuación nos dispondremos a visualizar algunas estadísticas de nuestro sistema software a través de la aplicación *GitStats* [6] y el análisis de estadísticas de código incluido dentro de *Ruby On Rails*.

En la tabla 6.1 podemos ver las estadísticas de código clasificadas según extensión y ordenados por el número de líneas que conforman la totalidad del proyecto.

Extensión	**Ficheros(%)**	**Líneas**	**Líneas / fichero**
pdf	1 (0,27 %)	13.850	13.850
png	41 (11,23 %)	7.573	184
rb	129 (45,54 %)	5.129	39
tex	14 (3,84 %)	3.662	261
yml	20 (5,48 %)	1.282	64
erb	46 (12,60 %)	1.089	23
dia	29 (7,95 %)	285	9
rake	3 (0,82 %)	259	86
sty	3 (0,82 %)	251	83
jpg	1 (0,27 %)	176	176
lock	1 (0,27 %)	158	158
scss	6 (1,64 %)	88	14
html	3 (0,82 %)	77	25
coffee	2 (0,55 %)	76	38
bib	1 (0,27 %)	41	41
gif	1 (0,27 %)	9	9

Tabla 6.1: Estadísticas del repositorio

En la tabla 6.2 podemos ver un análisis del código fuente según la estructura de un proyecto realizado en *Ruby On Rails*.

Nombre	Líneas	LOC	Clases	Métodos	M/C	LOC/M
Controladores	642	519	10	56	5	7
Helpers	274	241	0	27	0	6
Modelos	1.897	1285	16	165	10	5
Librerías	188	174	0	9	0	17
Test de integración	0	0	0	0	0	0
Test funcionales	346	273	11	0	0	0
Test unitarios	388	121	25	1	0	119
Total	3.735	2.613	62	258	4	8

Tabla 6.2: Estadísticas del código fuente Rails

A continuación pasamos a describir los términos que aparecen en la figura 6.2 para una mejor interpretación de la misma:

- Líneas: Son las líneas de código totales de cada una de las categorías.
- LOC: Son las líneas de código reales no autogeneradas por el framework.
- Clases: Son el número de clases pertenecientes a cada una de las categorías.
- Métodos: El número de métodos correspondientes a cada una de las categorías.
- M/C: Media de métodos por clase.
- LOC/M: Media de número de líneas por método.

6.3. Herramientas de desarrollo

6.3.1. Codificación

Para la creación y edición de los ficheros fuente del proyecto hemos utilizado el editor de texto gratuito *SublimeText2* [8], el cual es lo bastante extensible para ser utilizado como un completo *IDE* de desarrollo para *Ruby On Rails*.

Haremos intensivo uso de sus características avanzadas de editor de texto y de autocomplección de código, así como de la característica de inserción rápida de texto a través de los llamados *snippets* [9].

Otra característica avanzada de este editor de texto extensible es su integración con *GIT* y cualquier tecnología a través del uso de plugins. Además de ser multiplataforma lo cual es idóneo para la depuración de código *CSS* en distintas plataformas.

6.3.2. Depuración

Para la depuración del código fuente usamos de forma extensiva el navegador web *Firefox* [11] el cuál nos proporciona la extensión *Firebug* [3].

Firebug es una extensión de *Firefox* creada y diseñada especialmente para desarrolladores y programadores web. Es un paquete de utilidades con el que se puede analizar (revisar velocidad de carga, estructura *DOM*), editar, monitorizar y depurar el código fuente, *CSS*, *HTML* y *JavaScript* de una página web de manera instantánea e inline.

Firebug no es un simple inspector como *Dom Inspector*, además edita y permite guardar los cambios, un paso por delante del conocido *Web Developer*. Su atractiva e intuitiva interfaz, con solapas específicas para el análisis de cada tipo de elemento (consola, *HTML*, *CSS*, *Script*, *DOM* y red), permite al usuario un manejo fácil y rápido. *Firebug* está encapsulado en forma de plug-in o complemento de *Mozilla*, es *Open Source*, libre y de distribución gratuita [19].

CAPÍTULO 7

Pruebas

La fase de pruebas es una de las partes más importantes del desarrollo software [12]. El objetivo de las pruebas software es la verificación de que el proyecto cumple con los requisitos especificados inicialmente cuando se comenzó el desarrollo. Según la metodología clásica de desarrollo software existen diferentes enfoques a la hora de realizar las pruebas de software, siendo todos ellos complementarios entre sí.

En este capítulo vamos a describir las pruebas que hemos que realizado en nuestro software para asegurar la corrección del mismo.

7.1. Tipos de Pruebas

Este proyecto es una mezcla entre un videojuego online y una aplicación de gestión. Por ello deberemos de hacer varios enfoques a la hora de hacer las pruebas de nuestro sistema software, a saber:

- Pruebas unitarias: Son aquellas que nos permiten probar el correcto funcionamiento de un módulo de código. De esta forma conseguimos saber que cada uno de los módulos que integrar el sistema software funcionan correctamente por separado. [21]
- Pruebas de integración: Son aquellas que se realizan en el ámbito del desarrollo software una vez que han sido aprobadas las pruebas unitarias. Se refieren a la prueba de todos los elementos unitarios que componen un proceso realizadas en conjunto. De esta forma conseguimos verificar que las partes de un sistema software funcionan de forma conjunta. [22]
- Pruebas funcionales: Son pruebas basadas en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software [23].

- Pruebas de usabilidad: Son aquellas pruebas que se encargan de medir cómo de bien puede una persona usar un sistema hecho por el hombre, como en nuestro caso es la página web. Consisten en seleccionar un grupo de usuarios y solicitarles que lleven a cabo las tareas para las cuales fue diseñada la aplicación, mientras que el equipo de desarrollo toma nota para evaluar la respuesta del usuario [13].

El primer grupo de pruebas pueden realizarse de forma automatizada, sin embargo, para la realización de las pruebas del segundo grupo será necesario la intervención de un grupo de usuarios que interactúen con la aplicación y que contesten un cuestionario para recopilar la información necesaria para la mejora de la aplicación.

7.2. Pruebas unitarias

Durante la fase de codificación perteneciente al capítulo 6 se fueron realizando al unísono las pruebas unitarias de cada componente software de forma no automatizada, evitando de esta forma en la medida de lo posible que en la fase de integración aparecieran errores referentes a la codificación.

Ruby On Rails proporciona un completo *framework* para el desarrollo de pruebas unitarias, de hecho, por cada modelo de nuestro sistema software, dentro del directorio `test/unit` se crea un fichero homónimo para la realización de pruebas unitarias. En nuestro caso, y debido a la necesidad de cumplir con el calendario de entrega del proyecto, no hemos hecho uso del mismo.

Una herramienta muy útil a la hora de desarrollar nuestro software y evitar errores en el código ha sido el uso del programa *SublimeText2* con el plugin *IntelLinter*. Este plugin es una herramienta que analiza las librerías y archivos del proyecto software y detecta un conjunto de errores sintácticos a través del análisis del código fuente.

7.3. Pruebas de integración

Según se iban desarrollando las diferentes funcionalidades del proyecto y estos iban cumpliendo los requisitos software recopilados en el capítulo 4 a través de las pruebas unitarias de cada uno de de los módulos que íbamos completando comprobamos que la integración de los mismos no repercutían en el software haciendo que se incumpliera alguno de los requisitos recolectados en la fase de análisis (Capítulo 4).

7.4. Pruebas funcionales

Una vez que habíamos comprobado que la integración de los módulos no remitía ningún tipo de error ni comportaba ningún tipo de comportamiento extraño en los módulos ya testeados con sus pruebas unitarias nos dispusimos a comprobar que el software desarrollado cumplía con los requisitos funcionales recopilados en la fase de análisis en el capítulo 4.

7.5. Pruebas de usabilidad

Para este proyecto se ha puesto especial énfasis en que la usabilidad de la web sea sencilla para cualquier tipo de usuario y para que además se visualice de forma correcta en la mayoría de navegadores web del mercado. Para ello se ha hecho uso del *framework CSS Twitter Bootstrap* [16].

Las características de este *framework CSS* son varias, entre ellas:

- **Multiplataforma**: Compatible con los navegadores web más actuales, ya sean móviles, tablets o de escritorio, inclusive para *Internet Explorer 7*.
- **Tabla de 12 columnas**: Trae predefinida una plantilla de 12 columnas que puede ser modificada.
- **Diseño Responsivo**: Los componentes de la página web se escalan según el tamaño del navegador y la resolución a la que son visualizados los mismos.
- **Guía de estilos documentada**: Es de los poco *frameworks CSS* que poseen una guía de estilos bien documentada por sus creadores para la modificación de la plantilla.
- **Plugins jQuery personalizados**: De esta forma cubrimos las necesidades más habituales en el desarrollo de una página web. Alertas, ventanas modales, popups, pestañas, etc...
- **Desarrollado con LESS/SASS:** Aunque *Twitter Bottstrap* fue desarrollado para su perfecta integración con *LESS*, dado que *Rails* es compatible de forma nativa con *SASS* las características de la definición de reglas se han migrado a este último. De esta forma podemos aprovechar las cualidades de anidado, *mixins*, declaración de variables, etc...

En las figuras 7.1 y 7.2 podemos ver la característica de diseño responsivo con la plantilla original de *Twitter Bootstrap*

Figura 7.1: Twitter bootstrap maximizado en Firefox

Figura 7.2: Twitter bootstrap minimizado en Firefox

7.5.1. Casos de pruebas con usuarios reales

Dado que la aplicación es web, se prevé que a ella accedan un gran número de usuarios. Dado que el deporte al que está orientado el software es el fútbol, se prevé que la edad de los posibles usuarios sea de cualquier rango de edades, desde infantes hasta personas adultas.

Por ello hemos seleccionado un conjunto de sujetos de prueba que abarquen diversas edades y sexo para conseguir una información lo suficientemente representativa y variada para perfeccionar nuestro sistema.

Gracias a estas pruebas hemos conseguido valiosa información para ir haciendo modificaciones al producto para hacerlo más completo y entretenido para el usuario.

Los usuarios que han realizado los casos de prueba son los siguientes:

Sujeto 1 Hombre, 15 años, bachillerato de la rama de ciencias sociales. Nivel de informática usuario alto. Conocimientos en ofimática y navegación web, mucha experiencia en videojuegos y redes sociales, alto nivel del panorama futbolístico.

Sujeto 2 Mujer, 20 años, estudios universitarios de la rama de letras. Nivel de informática usuario, alto. Conocimientos en ofimática, navegación web, redes sociales y dispositivos móviles, bajo nivel del panorama futbolístico.

Sujeto 3 Hombre, 25 años, estudios no universitarios. Nivel de informática usuario, alto. Conocimientos en ofimática, navegación web, redes sociales, desarrollo web. Conocimiento del panorama futbolístico, medio.

Sujeto 4 Hombre, 35 años, estudios universitarios de la rama de ciencias. Alto conocimiento en ofimática, desarrollo software, navegación web. Conocimiento alto del panorama futbolístico.

Sujeto 5 Hombre, 55 años, estudios universitarios de la rama de ciencias. Conocimiento medio de ofimática, bajo nivel de conocimiento en navegación web. Alto conocimiento del panorama futbolístico.

Los objetivos que han de pasar los usuarios son:

Objetivo 1 Ser capaz de registrarse, iniciar sesión y salir de la cuenta de usuario.

Objetivo 2 Ser capaz de crear su propia alineación para el siguiente partido y la táctica que desea utilizar.

Objetivo 3 Ser capaz de estipular el precio de las entradas para el siguiente partido disputado en su estadio.

Objetivo 4 Ser capaz de poner a entrenar a cualesquiera de los jugadores de su equipo desee y el entrenamiento que así fije.

Objetivo 5 Ser capaz de emitir, cancelar, recibir, aceptar y rechazar una oferta por uno de sus jugadores.

Objetivo 6 Ser capaz de visualizar las jornadas siguientes de la temporada actual en curso.

Para realizar estos casos de prueba por parte de los sujetos del estudio se contemplaron las siguientes variables:

- Todas las pruebas se realizaron bajo el navegador web Firefox 14 instalado bajo Windows 7.
- Se les proporciona un enlace web a la aplicación web alojada en un servidor remoto [1]
- No se les dio ninguna indicación de cómo navegar a través de la página web.

Los resultados de las pruebas realizadas según los objetivos propuestos se pueden observar en la tabla 7.1

	Objetivo 1	Objetivo 2	Objetivo 3	Objetivo 4	Objetivo 5	Objetivo 6
Sujeto 1	✓	✓	✓	✓	✓	✓
Sujeto 2	✓	✓	✓	✓	✓	✓
Sujeto 3	✓	✓	✓	✓	✓	✓
Sujeto 4	✓	✓	✓	✓	✓	✓
Sujeto 5	✓	✓	X	X	X	X

Tabla 7.1: Pruebas pasadas por los usuarios

Como podemos observar el único usuario con problemas para terminar de pasar las pruebas fue el Sujeto 5, no es de extrañar puesto que sus conocimientos informáticos son básicos y tiene poca intuición al haber usado muy poco aplicaciones web interactivas.

CAPÍTULO 8

Resumen

En este proyecto desarrollaremos un sistema web de gestión de clubes de fútbol orientado al disfrute del usuario como de si un videojuego se tratara. Este sistema nos ha permitido conocer los entresijos del desarrollo software web mediante el uso del *framework* de desarrollo rápido de aplicaciones web *Ruby On Rails*.

8.1. Introducción y objetivos del proyecto

Este proyecto se inició para completar la formación académica de la titulación sede Ingeniería Técnica en Informática de Gestión de la Escuela Superior de Ingeniería de Cádiz de la Universidad de Cádiz. Gracias a ella, pudimos aprender cómo desarrollar un producto totalmente funcional usando para ello el uso de un framework basado en el patrón *Modelo-Vista-Controlador Ruby On Rails*.

Además, comprendimos la dificultad que conlleva el desarrollo de un sistema software, desde la misma etapa de conceptualización del sistema.

El objetivo del proyecto es proporcionar un videojuego web basado en la gestión de clubes de fútbol que se adapte a las condiciones de licencia del software libre que sea capaz de competir en un futuro con las alternativas actualmente existentes en el mercado actual.

Al concluir el desarrollo, la aplicación ha de ser capaz de desempeñar todas las funcionalidades descritas en el capítulo 4.

8.2. Análisis del sistema

Para conocer los requisitos funcionales del sistema a desarrollar nos reunimos varias veces con nuestro tutor de proyecto en las primeras semanas de desarrollo del sistema.

De dichas reuniones, fuimos capaces de llegar a numerosas conclusiones sobre cuáles serían los límites y funcionalidades que habríamos de desarrollar en el proyecto, para poder crear la Especificación de Requerimentos (SRS) [15]

8.2.1. Características de los usuarios

El producto está dirigido a todo usuario que sepa manejar un navegador web y que esté interesado en la simulación de la gestión de un club de fútbol, así como conocer las reglas y premisas básicas de negocio del deporte.

Identificamos dos roles bien diferenciados en nuestro sistema:

1. **Usuario Jugador**: Se refiere a una persona que se registra en el portal web y que participará en la gestión de un club al que se le asignará. Este usuario podrá conectarse al sistema e interactuar con el mismo a través de la interfaz web.
2. **Usuario Administrador**: Es un tipo de usuario que será capaz de modificar el comportamiento del sistema a través del menú de administración de nuestra aplicación para llevar a cabo las acciones de gestión de la competición de nuestro sistema.

8.2.2. Requisitos de rendimiento

Este aspecto es fundamental en el análisis del sistema, se ha de tener en cuenta que la aplicación es de tipo servidor y que a ella estarán conectados concurrentemente un número de usuarios elevado.

Se ha considerado también el rendimiento que nos ofrece el diseño web realizado. Para que nuestra página web sea compatible con la mayoría de dispositivos y se ejecute lo suficientemente fluida hemos decidido no usar la tecnología propietaria *Flash*, usando para la renderización de efectos la librería *JQuery* en combinación con hojas de estilo *CSS*.

El rendimiento de la aplicación podrá ser menor en la primera instanciación de la aplicación debido a que no se habrá cacheado ningún contenido.

8.2.3. Requisitos de interfaces externas

Aquí detallaremos los requisitos de conexión a otros sistemas hardware o software con los que vamos a interactuar.

8.2.3.1. Interfaces de usuario

Podemos dividir los requisitos de interfaz de usuario según tres roles:

- **Usuario Jugador y Usuario Administrador:**
  - Interfaz atractiva para los usuarios: Es fundamental para la buena aceptación de nuestra aplicación el crear una interfaz lo suficientemente clara, dinámica y estética para que el usuario sienta una inmersión en la experiencia de juego. Una interfaz no atractiva sólo traerá problemas de rechazo y disconformidad por parte del usuario.

    Para que la interfaz sea sencilla de modificar y tratándose de una aplicación web usaremos *hojas de estilo CSS*, así todo el apartado de diseño gráfico podrá ser modificado posteriormente sin que por ello afecte al contenido ni la estructura del documento *HTML*.
  - Una característica de nuestra aplicación es que el sistema de avisos está embebido dentro de la página web y es muy parecido al sistema de notificación de los sistemas operativos *Mac OS X* y de la *distribución Linux Ubuntu*.
- **Superusuario:** En este caso la interfaz corresponderá a la que tenga el propio sistema operativo dónde esté instalado la aplicación.

8.2.3.2. Interfaz con el hardware

Las copias de seguridad del sistema se realizarán de forma periódica cada día, semana y mes de forma incremental y utilizando la herramienta rsync. Dichas copias de seguridad se almacenarán en otro servidor dedicado exclusivamente al almacenaje de las copias.

8.2.3.3. Interfaz con el software

La aplicación web correrá en el lado del servidor en un sistema *Linux* con *Ubuntu Server 12.04*. Mientras que en el lado cliente podrá ejecutarse en cualquier sistema que disponga de un navegador web que cumpla los estándares 2.0 w3c.

8.2.4. Requisitos de rendimiento

Este aspecto es fundamental en el análisis del sistema, se ha de tener en cuenta que la aplicación es de tipo servidor y que a ella estarán conectados concurrentemente un número de usuarios elevado.

Se ha considerado también el rendimiento que nos ofrece el diseño web realizado. Para que nuestra página web sea compatible con la mayoría de dispositivos y se ejecute lo suficientemente fluida hemos decidido no usar la tecnología propietaria *Flash*, usando para la renderización de efectos la librería *JQuery* en combinación con hojas de estilo *CSS*.

El rendimiento de la aplicación podrá ser menor en la primera instanciación de la aplicación debido a que no se habrá cacheado ningún contenido.

8.2.5. Requisitos de Información

El sistema almacenará en forma de tablas la siguiente información relativa a las siguientes entidades:

User Se almacenan los datos de cada uno de los usuarios que interactúan con la aplicación.

Club Se almacenan los datos de los clubes que participan en las ligas.

League Se almacenan los datos de las ligas que existen en el sistema.

Season Se almacenan los datos de las temporadas correspondientes a cada liga del sistema.

Round Se almacenan los datos de las jornadas de cada temporada.

MatchGeneral Se almacenan los partidos que se han de jugar cada jornada.

MatchDetail Se almacenan las acciones que tienen lugar en cada partido.

LineUp Se almacenan las alineaciones de cada equipo y partido.

ClubFinancesRound Se almacenan los datos de finanzas correspondientes a cada jornada y club.

Player Se almacenan los datos de los jugadores de fútbol de los clubes.

Offer Se almacenan los datos de las ofertas que se realizan los usuarios de la aplicación sobre los jugadores de fútbol.

Training Se almacenan los datos de los entrenamientos de los jugadores de fútbol.

AdminMessages Se almacenan los mensajes de administración hacia los usuarios.

8.3. Diseño del sistema

Para el diseño del sistema tuvimos que tomar algunas decisiones básicas, sobre todo orientadas a qué tecnologías emplear.

Entre ellas destacamos el uso de la tecnología de Software Libre *Ruby On Rails*, basado en el lenguaje de *script Ruby*. Dicho *framework* nos proporciona todas las herramientas necesarias para el desarrollo del sistema incluyendo el de la interfaz cliente.

8.3.1. Arquitectura

La arquitectura que emplearemos es la que mejor se ajusta a las aplicaciones cliente/servidor, aplicando el patrón *Modelo-Vista-Controlador* que nos proporciona de base *Rails* [14].

El patrón *Modelo-Vista-Controlador* divide el software en tres capas bien diferenciadas que se adaptan muy bien a la tecnología Cliente-Servidor. Dichas capas son:

- **Modelo**: Esta capa es la relativa a la representación de la información con la que el proyecto interactuará. Está compuesto por las clases necesarias para el acceso, modificación, búsqueda y demás operaciones con los datos.

- **Vista**: En esta capa agruparemos todas las clases necesarias para la representación de la interfaz con la que nuestros usuarios interactuarán.

- **Controlador**: Será la capa que nos permita gestionar las peticiones que lleguen a nuestro servidor.

Los modelos usan el motor de persistencia *ActiveRecord* integrado en *Rails* y basado en el patrón de mismo nombre *Active Record* [18]. Esta clase es la que en última instancia ejecuta las órdenes de escritura y lectura en lenguaje *SQL* en el *SGBD* que el administrador haya decidido usar, en nuestro caso *SQLite3*.

La vista son los ficheros con extensión `.erb` basados en el sistema de plantillas *eRuby*.

8.3.2. Diagrama de componentes

Podemos ver el diagrama de componentes en la figura 8.1.

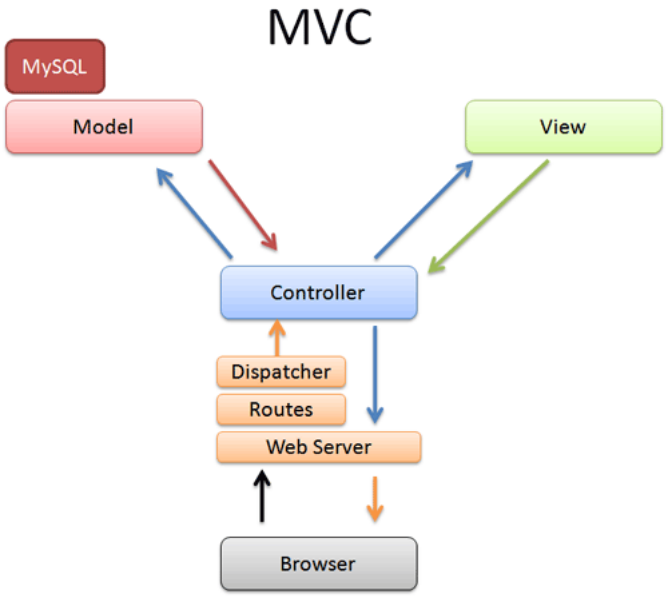

Figura 8.1: Diagrama de componentes

El framework *Ruby on Rails* hace especial hincapié en tres conceptos muy básicos que caracterizan toda su estructura, que son:

- **Convention over Configuration (COC)**: Convención frente a configuración.
- **Don't Repeat Yourself (DRY)**: No te repitas
- **Keep It Simple (KIS)**: Mantenlo simple.

Gracias a ello seguiremos la estructura de directorios en árbol para nuestro código fuente que genera *Rails* para una aplicación vacía.

8.3.3. Entorno de ejecución

En la figura 8.2 podemos ver el diagrama de despliegue de la aplicación puesta en marcha en cualquiera de los entornos, ya sea este de desarrollo, test o producción.

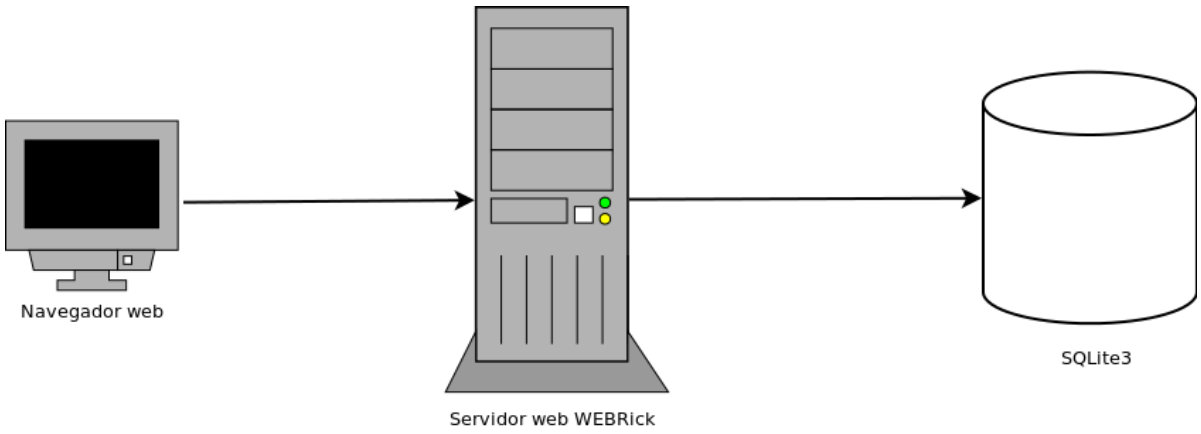

Figura 8.2: Diagrama de estructura

El sistema (servidor) interactuará con uno o más clientes web (navegadores). El servidor, a su vez estará formado por un servidor de aplicaciones web, en nuestro caso *Webrick*, y el sistema de gestión de base de datos será *SQLite3*.

Gracias a esta sencilla configuración no tendremos que configurar ninguna cuenta ni servicio en nuestro servidor web.

8.3.4. Diagramas de clases conceptuales

A continuación mostramos algunos de los modelos de clases conceptuales en notación *UML* [10]

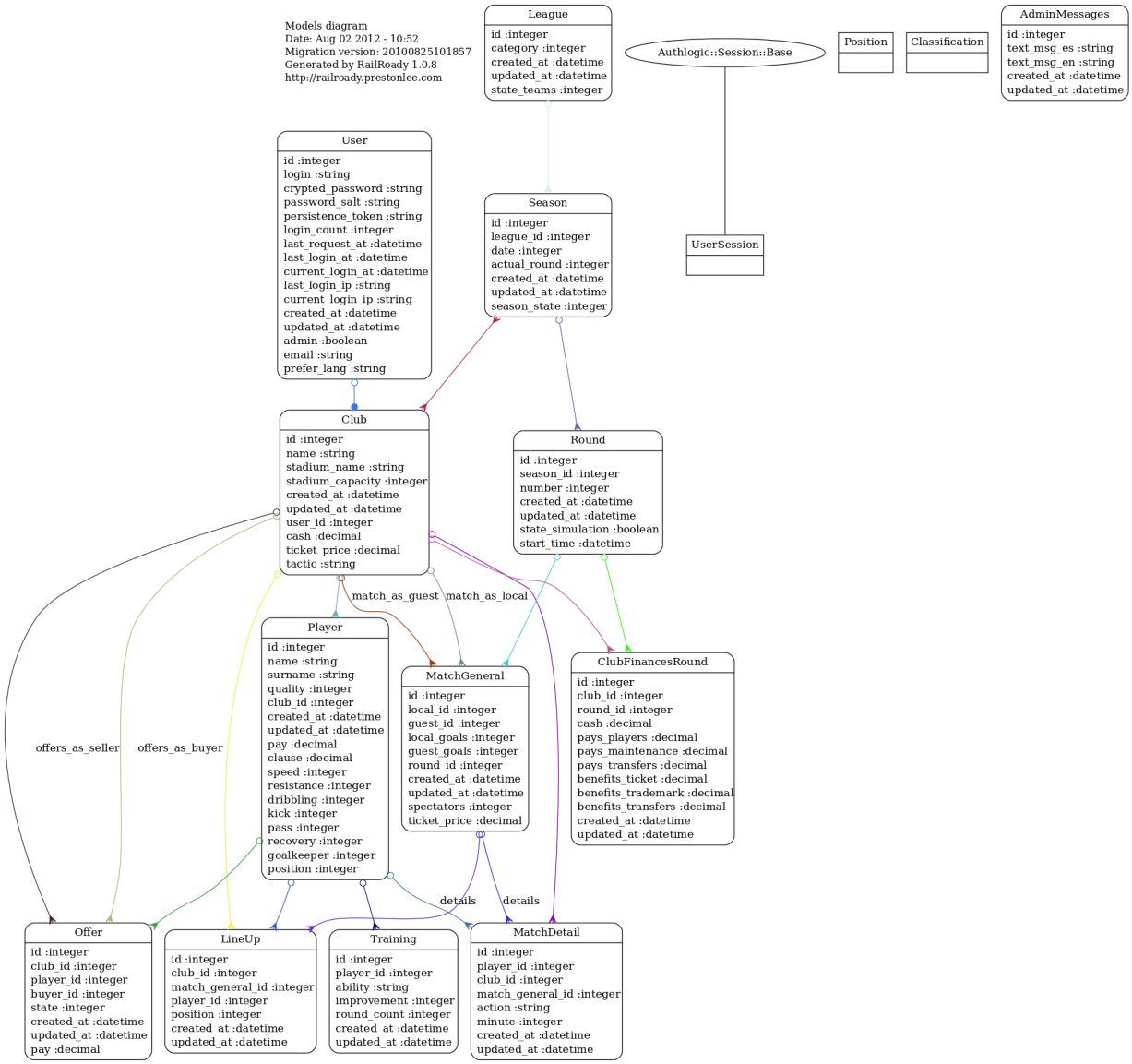

Figura 8.3: Diagrama de Clases UML de la capa modelo

Figura 8.4: Diagrama de Clases UML de la capa controlador

8.4. Implementación

Para la implementación del software hemos utilizado los siguientes lenguajes:

- Ruby como lenguaje servidor.
- HTML como lenguaje de marcado.
- CoffeScript como metalenguaje para el desarrollo en *JavaScript*[7].
- SASS como metalenguaje para el desarrollo de hojas de estilo [5].

Y las aplicaciones utilizadas para codificar a de una forma sencilla y productiva:

- El editor de texto extensible *SublimeText2* [8].
- El depurador de *Firefox* [11] *Firebug*. [3].
- El controlador de versiones *GIT* [20].
- El *framework CSS TwitterBootstrap*. [16].

8.5. Pruebas y validación

En la fase de análisis comprobamos la validez de nuestros requisitos identificando de forma correcta cuáles serían las funcionalidades, restricciones y el ámbito de desarrollo y ejecución de nuestra aplicación; además de comprobar que dichos requisitos no se contradicen ni acoplan.

Posteriormente validamos que la especificación de requisitos era correcta con nuestro tutor de proyecto de fin de carrera.

Las pruebas efectuadas sobre el entorno de desarrollo han sido de tipo manual en todo momento debido a la necesidad de cumplir con el calendario de entrega de las aplicación.

8.6. Conclusiones y trabajo futuro

Este proyecto ha servido para poder completar la formación académica recibida en la Escuela Superior de Ingeniería orientada a la finalización de la titulación de Ingeniería Técnica en Informática de Gestión de la Universidad de Cádiz.

El desarrollo de este sistema software nos ha permitido conocer en profundidad cómo se realiza el desarrollo de una aplicación a través de la utilización de un *framework* y las dificultades y retos que esto conlleva.

Este sistema es altamente ampliable debido a la naturaleza de videojuego del mismo. Una de las características a mejorar y ampliar sería la de la creación de un sistema de simulación de partidos basadas en tecnologías avanzadas de inteligencia artificial, lo cual llevaría asociada una plena experiencia de usuario.

CAPÍTULO 9

Conclusiones y trabajo futuro

Para terminar vamos a desarrollar este capítulo dónde hablaremos de nuestras expectativas futuras en torno a la posible extensión del sistema software creado y las conclusiones que hemos sacado a través del trabajo que hemos realizado en torno al mismo.

9.1. Conclusiones Generales

Como resultado del desarrollo del sistema *Simulator Football Online* hemos sido capaces de desarrollar una aplicación web multidioma y multijugador que será capaz de poner en contacto a diferentes usuarios para simular la gestión que se realizan en los clubes de fútbol desde el entrenamiento de los jugadores que conforman al equipo hasta la estipulación del precio de las entradas del siguiente partido.

El resultado que hemos obtenido al término del desarrollo del sistema software es el de un producto web al que podrán acceder un número ilimitado de personas dada la naturaleza de libre acceso con la que ha sido diseñado. Gracias a ello se han colmado varias expectativas personales y profesionales.

9.1.1. Primer simulador de la gestión de clubes de fútbol libre desarrollado en Ruby On Rails

Simulator Football Online es el primer simulador de la gestión de clubes de fútbol realizado sobre *Ruby On Rails* licenciado mediante la fórmula de software libre, en concreto, liberado bajo licencia *GPL.*

Esto supone varias ventajas que otros sistemas no aportan, sobre todo las inherentes a su naturaleza libre, entre ellas destacamos [4]:

- Libertad de usar el programa con cualquier propósito. Aunque distribuyamos esta primera versión de manera gratuita, no habría ningún problema en hacer una explotación comercial del sistema.
- Libertad para estudiar cómo funciona el programa y modificarlo, adaptándolo a cualquier necesidad. Esto supondría que podrían crearse varios *forks* para adaptarlo a otro tipo de competiciones deportivas.
- La libertad para distribuir copias del programa, con lo que gracias a ello se puede llegar a un mayor número de usuarios.
- La libertad para mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie de ello. Gracias a esto, posibles errores serían modificados con mayor celeridad.

El licenciar este programa con una licencia de software libre supone devolver a la comunidad todo lo que ella me ha aportado a mi. Sólo hace falta visionar en el capítulo 6 el número de herramientas con licencia de software libre para darse cuenta que de no disponer de la licencia de software libre pertinente, este proyecto no podría haberse llevado a cabo.

9.1.2. Aplicación de lo aprendido durante la etapa educativa

Este proyecto no podría haber sido llevado a buen término de no haber cursado numerosas asignaturas de la carrera universitaria cursada, a destacar:

Ingeniería del Software Impartida por Doña Elena García Orta. En ella aprendimos todo lo necesario para poder hacer un análisis y diseño software que hiciera la etapa de codificación prácticamente un mero trámite.

Programación en Internet Impartida por Don Manuel Palomo Duarte y actual tutor de este proyecto de fin de carrera. Gracias a la asignatura se sentaron las bases de cómo funciona la tecnología web, sus protocolos. Sin ella hubiera sido mucho más complicado el acercamiento al desarrollo de un portal web multiusuario.

Inteligencia Artificial Impartida por Don Ignacio Pérez Blanquer. Gracias a ella supimos evaluar la forma más sencilla para realizar un algoritmo lo suficientemente rápido y realista para la consecución de una simulación más o menos real del desarrollo de un partido de fútbol.

Comercio electrónico Aún acudiendo a esta asignatura como oyente, al pertenecer a la titulación de Ingeniería Informática, esta asignatura fue impartida en su primer año por Don Juan Manuel Dodero Beardo y precisamente utilizando como *framework* el que hemos utilizado en este proyecto, *Ruby On Rails*.

De todas formas, aún siendo estas asignaturas las que más competencias académicas tenían en relación con el proyecto, no hay que menospreciar la formación y conocimientos aportadas por todas las que componen la carrera como son, Programación Orientada a Objetos, Programación Funcional, Sistemas Operativos, Álgebra, y tantas otras que nos han aportado no solo los conocimientos inherentes al temario de las asignaturas, sino a cómo pensar, aprender y ser autosuficientes.

9.1.3. Capacidad de iniciativa

Este proyecto fue una propuesta propia. Dado que pensaba que tenía que orientar el proyecto de fin de carrera a todo lo aprendido en la misma, y teniendo en cuenta que la carrera pertenece a al rama de gestión de la informática, quise orientar el desarrollo del proyecto al desarrollo de un sistema de gestión.

Teniendo en cuenta que los sistemas de gestión actuales suelen ir dirigidos a colmar las necesidades de un programa integral que abarque toda la planificación de recursos empresariales (*ERP*), quisimos distanciarnos un poco de ese tipo de sistemas y crear algo que nos reconfortara cada vez que estuviéramos desarrollándolo, pero siempre teniendo como objetivo poner en práctica los conocimientos adquiridos en la carrera.

Por ello propusimos a nuestro tutor de proyecto el desarrollo de un videojuego integrado como una aplicación web de gestión de clubes de fútbol.

9.1.4. Desarrollo íntegro

Es una gran satisfacción, tanto en el plano personal como en el profesional el desarrollo íntegro de un sistema software desde sus etapa de conceptualización hasta la de su publicación. Además, ha supuesto un gran reto el enfrentarnos a una tecnología relativamente nueva que hace un uso intensivo de la programación funcional debido a la característica del lenguaje del *framework* que hemos escogido para el desarrollo.

Dado que hemos desarrollado este proyecto de forma individual, ha sido necesario aplicar conocimientos y desarrollar algunas competencias de las que adolecíamos, a destacar:

Usabilidad y compatibilidad web Ha sido necesario aprender todo lo necesario respecto a usabilidad web para realizar una página web lo suficientemente sencilla y atractiva para que fuera usada por cualquier tipo de persona.

Debido a nuestras limitaciones como grafistas, tuvimos que buscar un sistema que nos proporcionara una forma sencilla de diseñar y realizar una página web accesible y que cumpliera con los suficientes estándares para que funcionara con un amplio rango de navegadores.

Conseguimos solventar nuestras carencias como grafistas gracias al uso del *framework CSS*, *Twitter Bootstrap*.

Planificación y Gestión de proyectos Dado que optamos por ser nosotros mismos quiénes delimitáramos los límites y ámbito de aplicación del proyecto tuvimos que embarcarnos en la necesidad de ser unos buenos gestores y planificadores del tiempo, así como prever cuáles serían los problemas que podría acarrear el desarrollo e implementación de algunas características a simple vista sencillas.

Esta fue una experiencia muy enriquecedora que nos reveló que no es nada sencillo el cumplir con el calendario proyectado para la implementación de una funcionalidad que hayamos decidido introducir si no se ha analizado de forma concienzuda las posibles contingencias que nos pueden surgir.

Herramientas de desarrollo Gracias a tener que hacer un desarrollo desde cero, tanto de la documentación como de la aplicación en sí, se nos hizo necesario investigar cuáles serían las herramientas que más se adecuaran a nuestro flujo de desarrollo. Gracias a ello descubrimos nuevas herramientas, como *SublimeText2* y aprendimos a utilizar nuevos controladores de la versión como es *GIT*, además de tener que acostumbrarnos a seguir un flujo de desarrollo estándar para comunicarnos con el cliente, que en este caso sería nuestro tutor de proyecto.

Aprendizaje de frameworks A lo largo del desarrollo tuvimos que afrontar la decisión de codificar la aplicación desde cero, o utilizar *frameworks*. Dado que uno de los objetivos que nos marcamos fue el aprendizaje de nuevas tecnologías nos decantamos por la utilización intensiva de *frameworks*, esto nos llevó a concluir que en un principio es un ligero escollo el desarrollo a través de un marco de trabajo ya preestablecido, ya que hay que hacer una inversión en forma de tiempo para conocer cómo funciona y se estructura, y adaptar de esta forma tus hábitos de programación a la estructura que nos proporciona el *framework* escogido.

9.2. Trabajos futuros

Dado que ha sido necesario imponer un límite de funcionalidades que desarrollar para cumplir con la fecha de entrega y presentación del proyecto dentro de unos límites razonables que se corresponden con las horas de estudio que representan los 6 créditos que conforman la matrícula del proyecto, muchas de las ideas que tuvimos a la hora de hacer la propuesta de proyecto fueron descartadas.

Hay que tener en cuenta la dificultad que supone que un sólo desarrollador sea el que lleve a cabo todas las etapas de desarrollo de una página web, hay que tener en cuenta además la dificultad añadida que supone que no tengamos muchos conocimientos en cuanto a diseño gráfico para hacer del producto algo mucho más atractivo de cara al usuario.

Debido a ello, lamentablemente, tuvimos que dejar fuera del pliego de especificación de requisitos algunas funcionalidades que creemos que harían del sistema desarrollado un simulador mucho más completo y atractivo para el usuario. Aún así, y debido a que vamos a liberar el sistema bajo una licencia de software libre, vamos a dejar constancia de cuáles son las características que creemos que pueden ser mejorables y cuáles funcionalidades no han sido siquiera implementadas, para que alguien, si lo desea, pueda implementarlas.

9.2.1. Sistemas de Inteligencia artificial

Creemos que la inteligencia artificial del sistema podría ser mejorada en varias secciones, a saber:

- Subsistema de simulación de sanciones.
- Subsistema de simulación de lesiones.
- Perfeccionamiento del algoritmo de simulación de partidos.
- Subsistema de simulación de mercado de agentes libres.

- Subsistema de simulación de vejez.
- Subsistema de simulación de junta directiva.

9.2.2. Personalización de perfil de usuario y equipo

Pensamos que sería muy interesante poder dejar al usuario la posibilidad de personalizar las características de su club de fútbol. Como pueden ser

- Posibilidad de cambiar el nombre del estadio de su equipo.
- Posibilidad de establecer un escudo para su equipo.
- Posibilidad de establecer un nombre para su equipo.

9.2.3. Sistema de estadísticas

Sería interesante el desarrollar una sección web que mostrara estadísticas y comparativas entre usuarios, futbolistas y clubes para que los jugadores puedan perfeccionar su forma de gestionar los clubes y así conseguir una experiencia de usuario más plena.

Apéndices

APÉNDICE A

Manual de usuario

APÉNDICE B

Manual de Instalación

B.1. Suposiciones previas

Supondremos que el sistema dónde vamos a instalar la aplicación es una distribución *Ubuntu 12.04* y que el *SGBD* que utilizaremos será *SQLite3*.

B.2. Instalación

B.2.1. Preparación del sistema

Para configurar el sistema vamos a instalar un gestor de versiones de *Ruby* llamado *RVM* [17], gracias a este gestor, el sistema podrá tener diferentes versiones de interpretes *Ruby*.

Para ello primero instalemos los siguientes paquetes:

```
sudo aptitude install build-essential git-core curl
```

Posteriormente nos descargaremos la versión más reciente de *RVM*

```
curl -L https://get.rvm.io | bash -s stable
```

Esto descargará un *script* en *bash* que será ejecutado automáticamente, clonando el repositorio alojado en *GitHub*.

Para que *RVM* sea reconocido sin problemas por nuestra terminal ejecutaremos el siguiente comando:

```
echo '[[ -s "$HOME/.rvm/scripts/rvm" ]] && . "$HOME/.rvm/scripts/rvm" # Load
RVM function' && ~/.bash_profile
```

Por último habremos de instalar unos últimos paquetes paras asegurarnos cumplir con las dependencias necesarias para instalar *Ruby 1.9.2*. Para ello ejecutaremos el siguiente comando:

```
sudo aptitude install build-essential bison openssl libreadline6
libreadline6-dev curl git-core zlib1g zlib1g-dev libssl-dev libyaml-dev
libsqlite3-0 libsqlite3-dev sqlite3 libxml2-dev libxslt-dev autoconf libc6-dev
ncurses-dev
```

Ya tenemos todas las dependencias necesarias instaladas, ahora sólo necesitaremos instalar la versión de *Rails* en el sistema, para ello ejecutaremos el siguiente comando:

```
rvm install 1.9.2
```

Una vez se termine de instalar modificaremos nuestro fichero personal de `.bashrc` añadiendo al final del fichero la siguiente línea:

```
source \$HOME/.rvm/scripts/rvm
```

Ahora ejecutaremos la siguiente línea para actualizar el sistema de gemas de *Ruby*

```
gem update --system
```

Ya tenemos el sistema preparado para instalar en nuestro sistema *Simulator Football Online*.

B.3. Instalación de Simulator Football Online

Lo primero que tendremos que hacer será bajarnos la última versión del repositorio de nuestra aplicación, para ello ejecutaremos el siguiente comando:

```
git clone git://github.com/paliyoes/pfc_sfo_rails3.git pfc_sfo
```

Una vez descargado deberemos ejecutar el siguiente comando dentro del directorio `pfc_sfo` de nuestro sistema:

```
mkdir doc log tmp
```

A continuación tendremos que instalar todas las dependencias de nuestro proyecto, para ello ejecutaremos el siguiente comando:

```
bundle install
```

En caso de no estar instalada la gema bundle, aprovecharemos para instalarla y volver a ejecutar el comando anterior.

```
gem install bundle
```

Ahora deberemos preparar la base de datos para que pueda correr en el servidor nuestra aplicación, para ello ejecutaremos los siguientes comandos:

```
rake db:create
rake db:migrate
```

Una vez terminado de ejecutar ambos comandos, podremos ejecutar un servidor con nuestra aplicación a través del comando:

```
rails s
```

Para acceder a la aplicación sólo tendremos que ir a nuestro navegador favorito y acceder a la ruta `http://localhost:3000`

APÉNDICE C

Difusión

Este proyecto ha obtenido difusión en los siguientes medios:

- Repositorio de *GitHub*, dónde está albergado todo el código, documentos y recursos generados para el proyecto [2].
- Incluido dentro de un subdominio dentro del sitio web *Heroku* [1]

APÉNDICE D

Fichero creación Base de Datos (schema.rb)

```
# encoding: UTF-8
# This file is auto-generated from the current state of the database. Instead
# of editing this file, please use the migrations feature of Active Record to
# incrementally modify your database, and then regenerate this schema definition.
#
# Note that this schema.rb definition is the authoritative source for your
# database schema. If you need to create the application database on another
# system, you should be using db:schema:load, not running all the migrations
# from scratch. The latter is a flawed and unsustainable approach (the more migrations
# you'll amass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended to check this file into your version control system.

ActiveRecord::Schema.define(:version => 20100825101857) do

  create_table "admin_messages", :force => true do |t|
    t.string   "text_msg_es", :null => false
    t.string   "text_msg_en", :null => false
    t.datetime "created_at",  :null => false
    t.datetime "updated_at",  :null => false
  end

  create_table "club_finances_rounds", :force => true do |t|
```

```
24    t.integer  "club_id",                                                    :null =>
          false
25    t.integer  "round_id",                                                   :null =>
          false
26    t.decimal  "cash",               :precision => 16, :scale => 2, :
          default => 0.0
27    t.decimal  "pays_players",       :precision => 16, :scale => 2, :
          default => 0.0
28    t.decimal  "pays_maintenance",   :precision => 16, :scale => 2, :
          default => 0.0
29    t.decimal  "pays_transfers",     :precision => 16, :scale => 2, :
          default => 0.0
30    t.decimal  "benefits_ticket",    :precision => 16, :scale => 2, :
          default => 0.0
31    t.decimal  "benefits_trademark", :precision => 16, :scale => 2, :
          default => 0.0
32    t.decimal  "benefits_transfers", :precision => 18, :scale => 2, :
          default => 0.0
33    t.datetime "created_at",
                                                                          :null =>
          false
34    t.datetime "updated_at",
                                                                          :null =>
          false
35  end
36
37  create_table "clubs", :force => true do |t|
38    t.string   "name",             :limit => 30,
                                                                  :null => false
39    t.string   "stadium_name",     :limit => 40,
                                                                  :null => false
40    t.integer  "stadium_capacity",                                          :
          null => false
41    t.datetime "created_at",

          :null => false
42    t.datetime "updated_at",

          :null => false
43    t.integer  "user_id"
44    t.decimal  "cash",                          :precision => 16, :scale
          => 2, :default => 0.0
45    t.decimal  "ticket_price",                  :precision => 4,  :scale
          => 2, :default => 0.0
46    t.string   "tactic",
                                                                 :default => "
          4-4-2", :null => false
47  end
48
49  create_table "clubs_seasons", :id => false, :force => true do |t|
50    t.integer "club_id"
51    t.integer "season_id"
52  end
```

```
53
54  create_table "leagues", :force => true do |t|
55    t.integer  "category",    :default => 1, :null => false
56    t.datetime "created_at",                 :null => false
57    t.datetime "updated_at",                 :null => false
58    t.integer  "state_teams", :default => 0
59  end
60
61  create_table "line_ups", :force => true do |t|
62    t.integer  "club_id",          :null => false
63    t.integer  "match_general_id", :null => false
64    t.integer  "player_id",        :null => false
65    t.integer  "position",         :null => false
66    t.datetime "created_at",       :null => false
67    t.datetime "updated_at",       :null => false
68  end
69
70  create_table "match_details", :force => true do |t|
71    t.integer  "player_id",        :null => false
72    t.integer  "club_id",          :null => false
73    t.integer  "match_general_id", :null => false
74    t.string   "action",           :null => false
75    t.integer  "minute",           :null => false
76    t.datetime "created_at",       :null => false
77    t.datetime "updated_at",       :null => false
78  end
79
80  create_table "match_generals", :force => true do |t|
81    t.integer  "local_id",
                                                    :null => false
82    t.integer  "guest_id",
                                                    :null => false
83    t.integer  "local_goals"
84    t.integer  "guest_goals"
85    t.integer  "round_id",
                                                    :null => false
86    t.datetime "created_at",
                                                  :null => false
87    t.datetime "updated_at",
                                                  :null => false
88    t.integer  "spectators",                          :default => 0
89    t.decimal  "ticket_price", :precision => 4, :scale => 2, :default =>
        0.0
90  end
91
92  create_table "offers", :force => true do |t|
93    t.integer  "club_id",
                                                    :null => false
94    t.integer  "player_id",
                                                  :null => false
95    t.integer  "buyer_id",
                                                   :null => false
96    t.integer  "state",                               :default => 0,
          :null => false
97    t.datetime "created_at",
                                                 :null => false
```

```
98    t.datetime "updated_at",
                                                        :null => false
99    t.decimal  "pay",        :precision => 16, :scale => 2, :default => 0.0
100 end
101
102 create_table "players", :force => true do |t|
103   t.string   "name",       :limit => 20,
                                                    :null => false
104   t.string   "surname",    :limit => 30,
                                                    :null => false
105   t.integer  "quality",
                                                                  :null
        => false
106   t.integer  "club_id"
107   t.datetime "created_at",
                                                                :null =>
        false
108   t.datetime "updated_at",
                                                                :null =>
        false
109   t.decimal  "pay",                     :precision => 8, :scale => 2
110   t.decimal  "clause",                  :precision => 8, :scale => 2
111   t.integer  "speed",                                                :
        default => 0, :null => false
112   t.integer  "resistance",                                           :
        default => 0, :null => false
113   t.integer  "dribbling",                                            :
        default => 0, :null => false
114   t.integer  "kick",                                                 :
        default => 0, :null => false
115   t.integer  "pass",                                                 :
        default => 0, :null => false
116   t.integer  "recovery",                                             :
        default => 0, :null => false
117   t.integer  "goalkeeper",                                           :
        default => 0, :null => false
118   t.integer  "position"
119 end
120
121 create_table "rounds", :force => true do |t|
122   t.integer  "season_id",                          :null => false
123   t.integer  "number",                             :null => false
124   t.datetime "created_at",                         :null => false
125   t.datetime "updated_at",                         :null => false
126   t.boolean  "state_simulation", :default => false
127   t.datetime "start_time"
128 end
129
130 create_table "seasons", :force => true do |t|
131   t.integer  "league_id",                   :null => false
132   t.integer  "date",                        :null => false
133   t.integer  "actual_round"
134   t.datetime "created_at",                  :null => false
135   t.datetime "updated_at",                  :null => false
136   t.integer  "season_state", :default => 0, :null => false
137 end
```

```
138
139   create_table "trainings", :force => true do |t|
140     t.integer  "player_id",    :null => false
141     t.string   "ability",      :null => false
142     t.integer  "improvement"
143     t.integer  "round_count"
144     t.datetime "created_at",   :null => false
145     t.datetime "updated_at",   :null => false
146   end
147
148   create_table "users", :force => true do |t|
149     t.string   "login"
150     t.string   "crypted_password"
151     t.string   "password_salt"
152     t.string   "persistence_token"
153     t.integer  "login_count"
154     t.datetime "last_request_at"
155     t.datetime "last_login_at"
156     t.datetime "current_login_at"
157     t.string   "last_login_ip"
158     t.string   "current_login_ip"
159     t.datetime "created_at",                          :null => false
160     t.datetime "updated_at",                          :null => false
161     t.boolean  "admin"
162     t.string   "email",              :default => "",   :null => false
163     t.string   "prefer_lang",        :default => "en"
164   end
165
166 end
```

Bibliografía

[1] Carlos Rafael Belizón Ibáñez. Aplicación web alojada en Heroku. http://pfc-sfo-rails3.herokuapp.com/.

[2] Carlos Rafael Belizón Ibáñez. Código fuente de Simulator Football Online. http://github.com/paliyoes/pfc_sfo_rails3.

[3] FirebugWorkingGroup. Página web oficial de Firebug. http://getfirebug.com/.

[4] Free Software Foundation. ¿Qué es Software Libre? http://www.gnu.org/philosophy/free-sw.es.html.

[5] Hampton Catlin. SASS. http://www.sass-lang.com.

[6] Heikki Hokkanen. Página oficial de GitStats. http://gitstats.sourceforge.net/.

[7] Jeremy Ashkenas. CoffeScript. http://www.coffeescript.org.

[8] Jon Skinner. Página oficial de SublimeText2.

[9] Jon Skinner. Snippets para Sublimetext. http://sublimetext.info/docs/es/extensibility/snippets.html.

[10] Kendall Scott Martin Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language (3erd Edition)*. Addison-Wesley Professional.

[11] Mozilla Foundation. Página web oficial de Mozilla Foundation. http://www.mozilla.org/.

[12] Glenford J. Myers. *The Art of Software Testing*. Wiley, 1979.

[13] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann, 1993.

[14] Dave Thomas y David Heinemeier Hansson Sam Ruby. *Agile Web Development with Rails, 3ed Edition*. The Pragmatic Programmers, 2010.

[15] Raymond Turner. The foundations of specification. 2004.

[16] Twitter Inc. Página web oficial Twitter Bootstrap. http://twitter.github.com/bootstrap/.

[17] Wayne E. Seguin. RVM: Ruby Version Manager. http://rvm.io/.

[18] Wikipedia. Active Record Pattern. http://en.wikipedia.org/wiki/Active_record_pattern.

[19] Wikipedia. Firebug. http://es.wikipedia.org/wiki/Firebug.

[20] Wikipedia. Git. http://es.wikipedia.org/wiki/Git.

[21] Wikipedia. Prueba unitaria. http://es.wikipedia.org/wiki/Prueba_unitaria.

[22] Wikipedia. Pruebas de integración. http://es.wikipedia.org/wiki/Pruebas_de_integración.

[23] Wikipedia. Pruebas funcionales. http://es.wikipedia.org/wiki/Pruebas_funcionales.

GNU Free Documentation License

Version 1.3, 3 November 2008

<http://fsf.org/>

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole

or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with … Texts.” line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.